C Dynamic Memory Allocation

CMPE 230 - Spring 2024

Gökçe Uludoğan

Library Functions

- `malloc()`
- `calloc()`
- `free()`
- `realloc()`

void *malloc(size_t size);

- Stands for “memory allocation”
- Allocates a block of memory of the specified number of bytes.
- Returns a **pointer of void** which can be casted into pointers of any form.

Syntax

```
ptr = (castType*) malloc(size);
```

Example

```
ptr = (float*) malloc(30 * sizeof(float));
```

void *calloc(int num, int size);

- Allocates memory and initializes all bits to zero.
- The size of elements is given as a parameter.

Syntax

```
ptr = (castType*) calloc(n, size);
```

Example

```
ptr = (float*) calloc(30, sizeof(float));
```

void free(void *address);

- Dynamically allocated memory must be released explicitly.

Syntax

```
free(ptr);
```

void *realloc(void *address, int newsize);

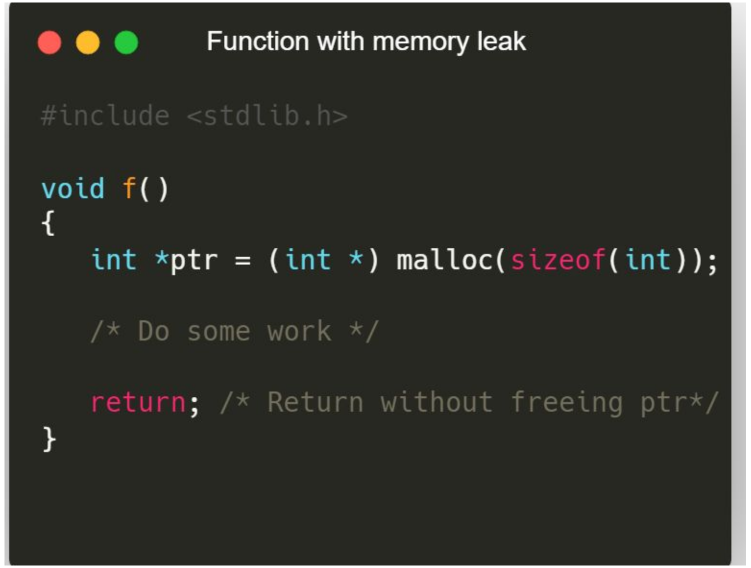
- Changes the size of previously allocated memory

Syntax

```
ptr = realloc(ptr, x);
```

Memory Leak

- Improper management of memory
- Occurs when memory which is no longer needed is not released
- To avoid memory leaks, memory allocated on heap should always be freed when no longer needed.


Function with memory leak

```
#include <stdlib.h>

void f()
{
    int *ptr = (int *) malloc(sizeof(int));

    /* Do some work */

    return; /* Return without freeing ptr*/
}
```

Memory Layout

Environment stores command line arguments, environment variables etc.

1. Stack
2. Heap
3. BSS (Uninitialized Data Segment)
4. DS (Initialized Data Segment)
5. Text

High	Environment
	Stack ↓
	. . . . . Empty
	Heap ↑
	BSS
	Data (DS)
Low	Text

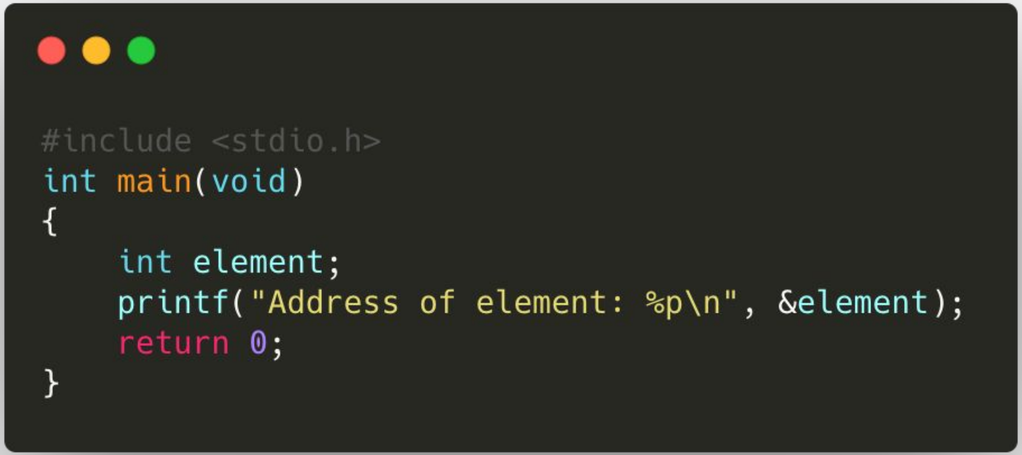
1. Stack

- Stack grows and shrinks opposite side of heap
- It stores temporary data such as function’s parameters, return address, and local variables.
- Each function has one stack frame.

High	Environment
	Stack ↓
	. . . . . Empty
	Heap ↑
	BSS
	Data(DS)
Low	Text

https://aticleworld.com/memory-layout-of-c-program/
https://stackoverflow.com/questions/10057443/explain-the-concept-of-a-stack-frame-in-a-nutshell

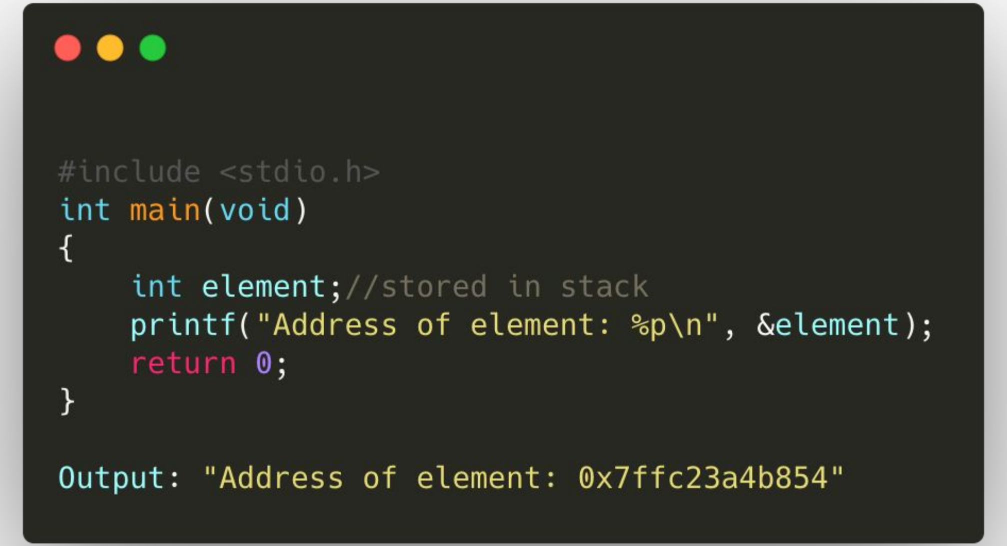
1. Stack



```c
#include <stdio.h>
int main(void)
{
    int element;
    printf("Address of element: %p\n", &element);
    return 0;
}
```

High	Environment
	Stack ↓
	. . . Empty . .
	Heap ↑
	BSS
	Data(DS)
Low	Text

1. Stack



```
#include <stdio.h>
int main(void)
{
    int element;//stored in stack
    printf("Address of element: %p\n", &element);
    return 0;
}

Output: "Address of element: 0x7ffc23a4b854"
```

High	Environment
	Stack ↓
	. . . Empty . .
	Heap ↑
	BSS
	Data(DS)
Low	Text

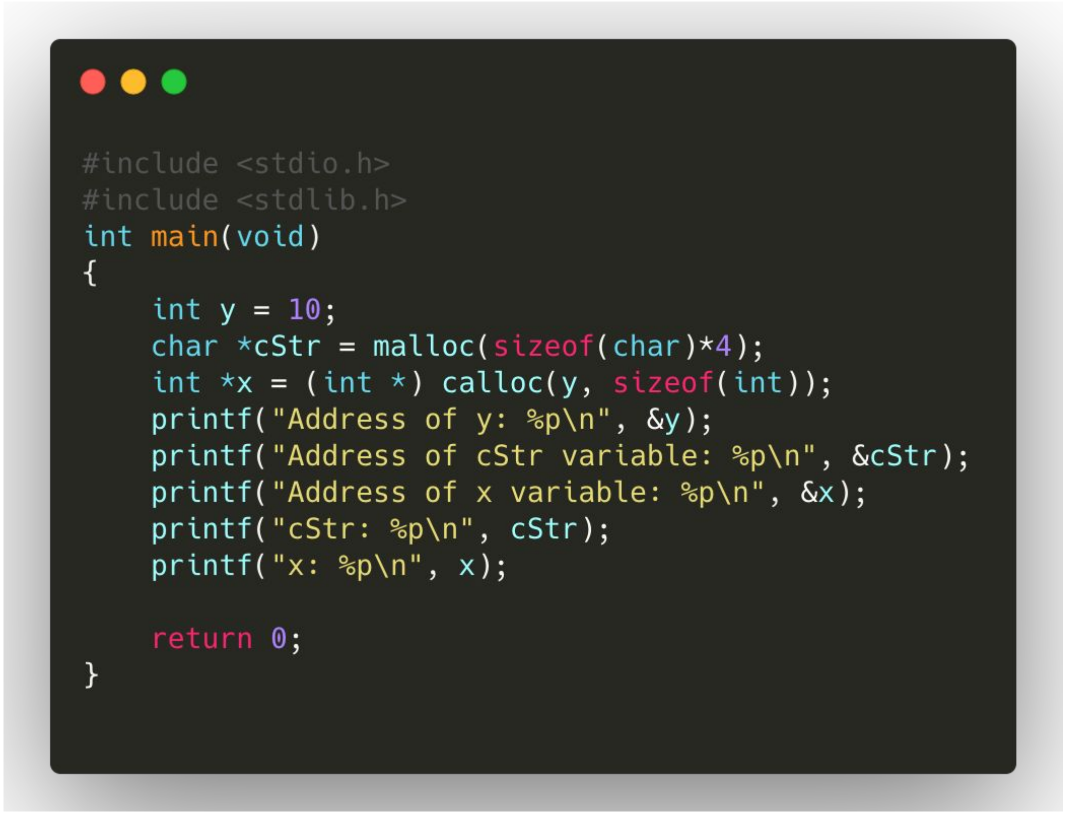
2. Heap

- Heap grows and shrinks opposite side of stack
- It allocates dynamic memory at run time with using these functions: malloc, calloc, free, etc

High	Environment
	Stack ↓
	. . . Empty . .
	Heap ↑
	BSS
	Data(DS)
Low	Text

https://aticleworld.com/memory-layout-of-c-program/

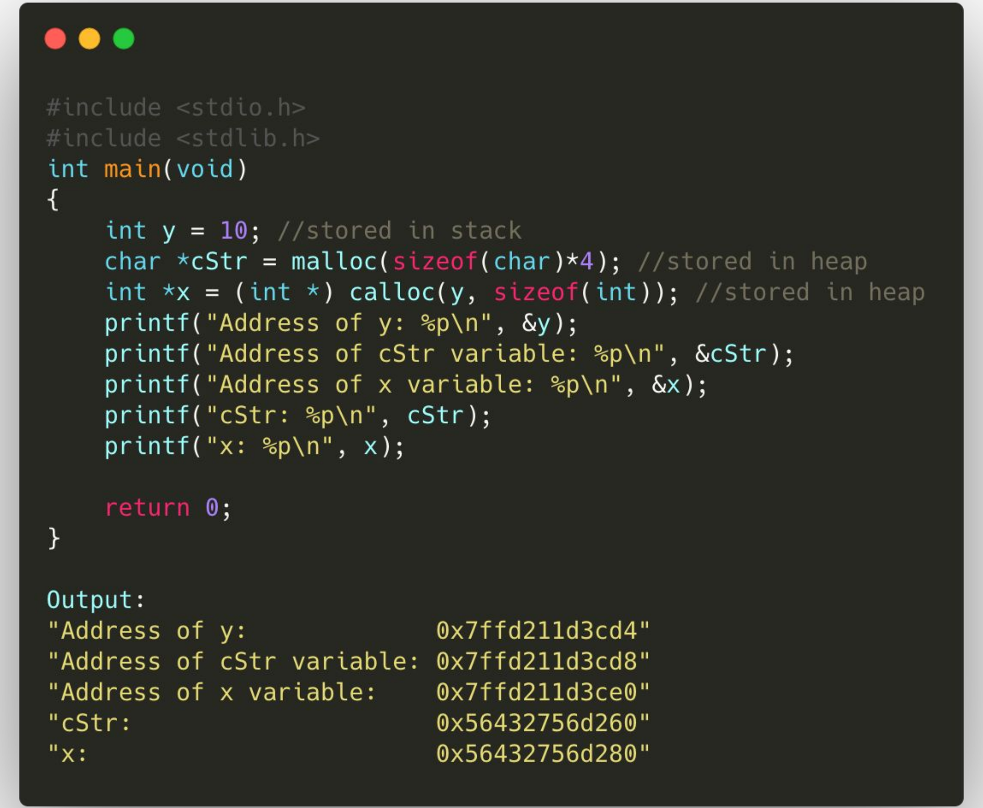
2. Heap



```
#include <stdio.h>
#include <stdlib.h>
int main(void)
{
    int y = 10;
    char *cStr = malloc(sizeof(char)*4);
    int *x = (int *) calloc(y, sizeof(int));
    printf("Address of y: %p\n", &y);
    printf("Address of cStr variable: %p\n", &cStr);
    printf("Address of x variable: %p\n", &x);
    printf("cStr: %p\n", cStr);
    printf("x: %p\n", x);

    return 0;
}
```

High	Environment
	Stack ↓
	. . . Empty . .
	Heap ↑
	BSS
	Data(DS)
Low	Text

2. Heap



```
#include <stdio.h>
#include <stdlib.h>
int main(void)
{
    int y = 10; //stored in stack
    char *cStr = malloc(sizeof(char)*4); //stored in heap
    int *x = (int *) calloc(y, sizeof(int)); //stored in heap
    printf("Address of y: %p\n", &y);
    printf("Address of cStr variable: %p\n", &cStr);
    printf("Address of x variable: %p\n", &x);
    printf("cStr: %p\n", cStr);
    printf("x: %p\n", x);

    return 0;
}

Output:
"Address of y:              0x7ffd211d3cd4"
"Address of cStr variable:  0x7ffd211d3cd8"
"Address of x variable:     0x7ffd211d3ce0"
"cStr:                      0x56432756d260"
"x:                         0x56432756d280"
```

High	Environment
	Stack ↓
	. . . Empty . .
	Heap ↑
	BSS
	Data(DS)
Low	Text

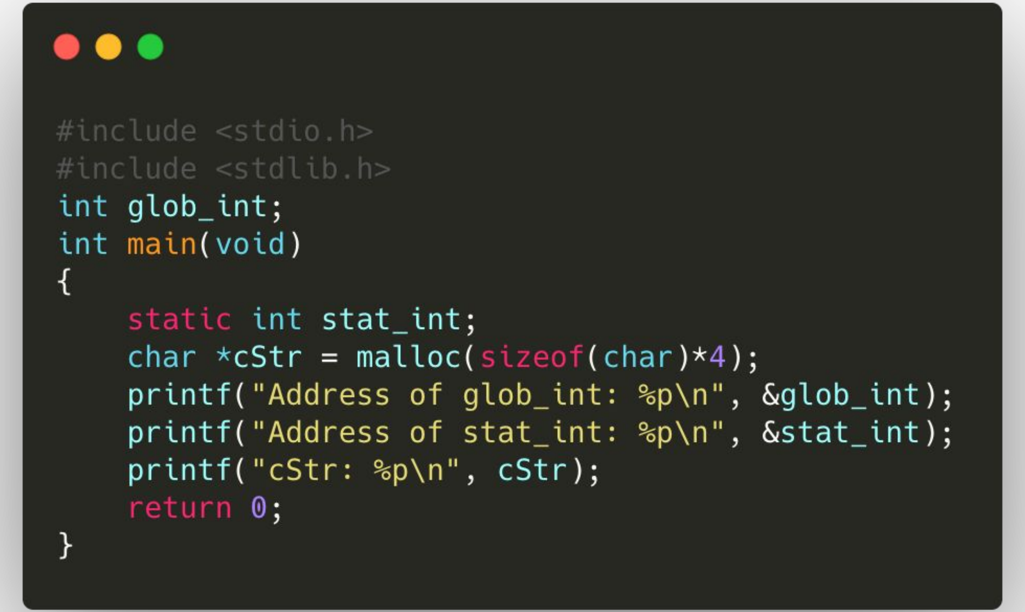
3. BSS

- It contains all uninitialized global and static variables.

High	Environment
	Stack ↓
	. . . Empty . .
	Heap ↑
	BSS
	Data(DS)
Low	Text

https://aticleworld.com/memory-layout-of-c-program/

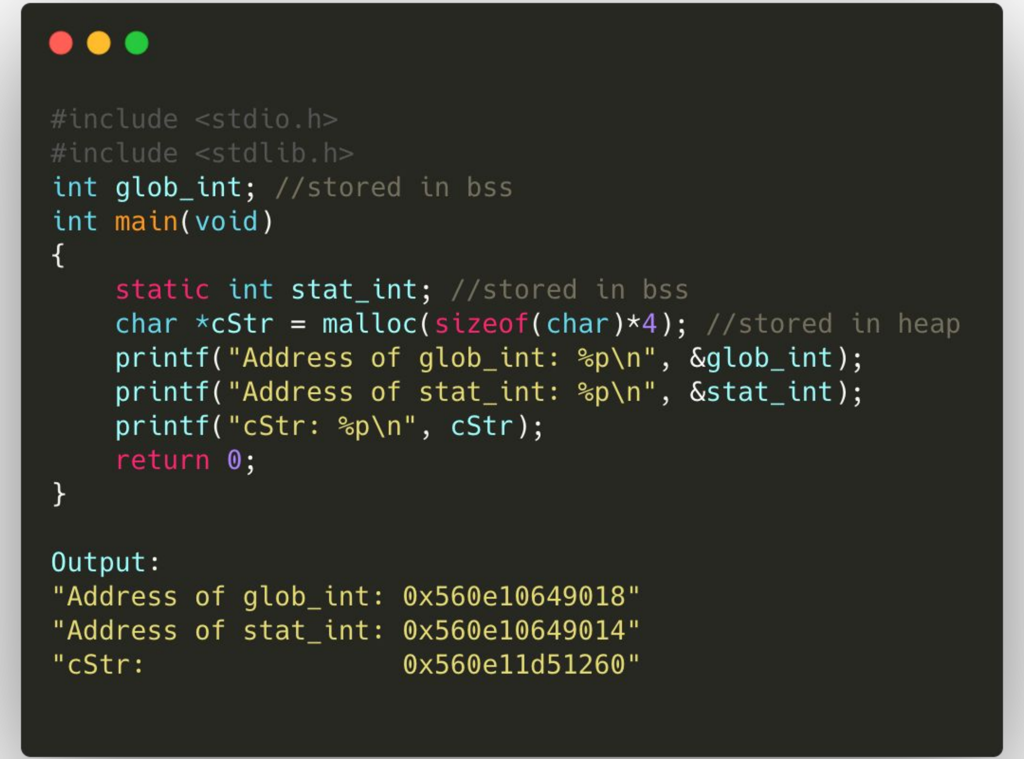
3. BSS



```c
#include <stdio.h>
#include <stdlib.h>
int glob_int;
int main(void)
{
    static int stat_int;
    char *cStr = malloc(sizeof(char)*4);
    printf("Address of glob_int: %p\n", &glob_int);
    printf("Address of stat_int: %p\n", &stat_int);
    printf("cStr: %p\n", cStr);
    return 0;
}
```

High	Environment
	Stack ↓
	. . . Empty . .
	Heap ↑
	BSS
	Data(DS)
Low	Text

3. BSS

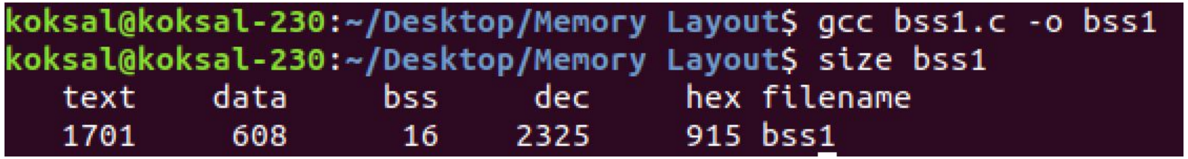


```c
#include <stdio.h>
#include <stdlib.h>
int glob_int; //stored in bss
int main(void)
{
    static int stat_int; //stored in bss
    char *cStr = malloc(sizeof(char)*4); //stored in heap
    printf("Address of glob_int: %p\n", &glob_int);
    printf("Address of stat_int: %p\n", &stat_int);
    printf("cStr: %p\n", cStr);
    return 0;
}

Output:
"Address of glob_int: 0x560e10649018"
"Address of stat_int: 0x560e10649014"
"cStr:                0x560e11d51260"
```

High	Environment
	Stack ↓
	. . . Empty . .
	Heap ↑
	BSS
	Data(DS)
Low	Text

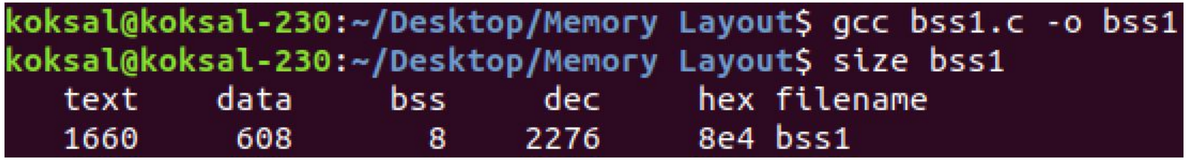
3. BSS

You can see size of BSS with this command:



```
koksal@koksal-230:~/Desktop/Memory Layout$ gcc bss1.c -o bss1
koksal@koksal-230:~/Desktop/Memory Layout$ size bss1
   text	   data	    bss	    dec	    hex	filename
   1701	    608	     16	   2325	    915	bss1
```

After removing stat_int:



```
koksal@koksal-230:~/Desktop/Memory Layout$ gcc bss1.c -o bss1
koksal@koksal-230:~/Desktop/Memory Layout$ size bss1
   text	   data	    bss	    dec	    hex	filename
   1660	    608	      8	   2276	    8e4	bss1
```

High	Environment
	Stack ↓
	. . . Empty . .
	Heap ↑
	BSS
	Data(DS)
Low	Text

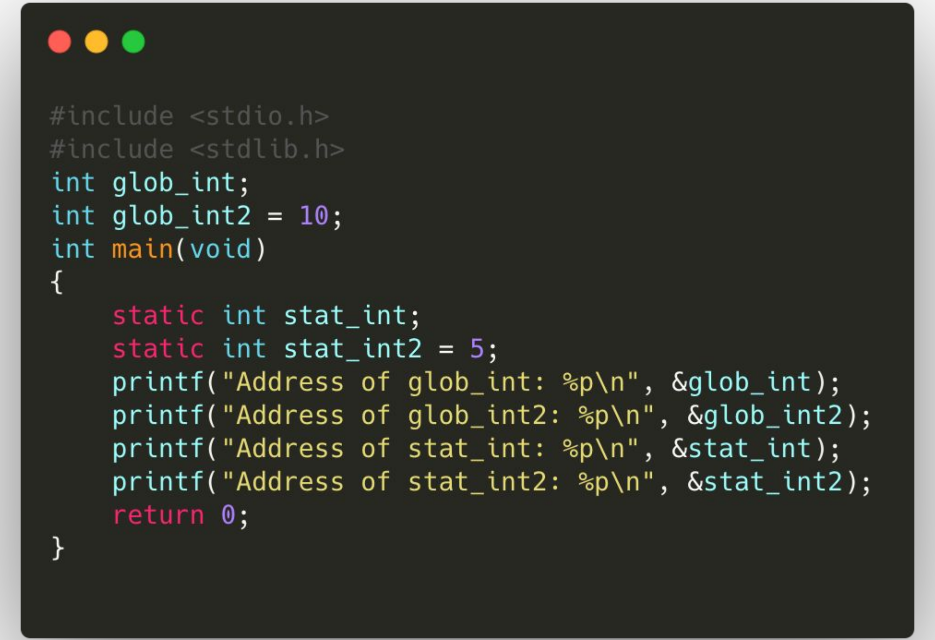
4. Data (DS)

- It contains explicitly initialized global and static variables.

High	Environment
	Stack ↓
	. . . Empty . .
	Heap ↑
	BSS
	Data(DS)
Low	Text

https://aticleworld.com/memory-layout-of-c-program/

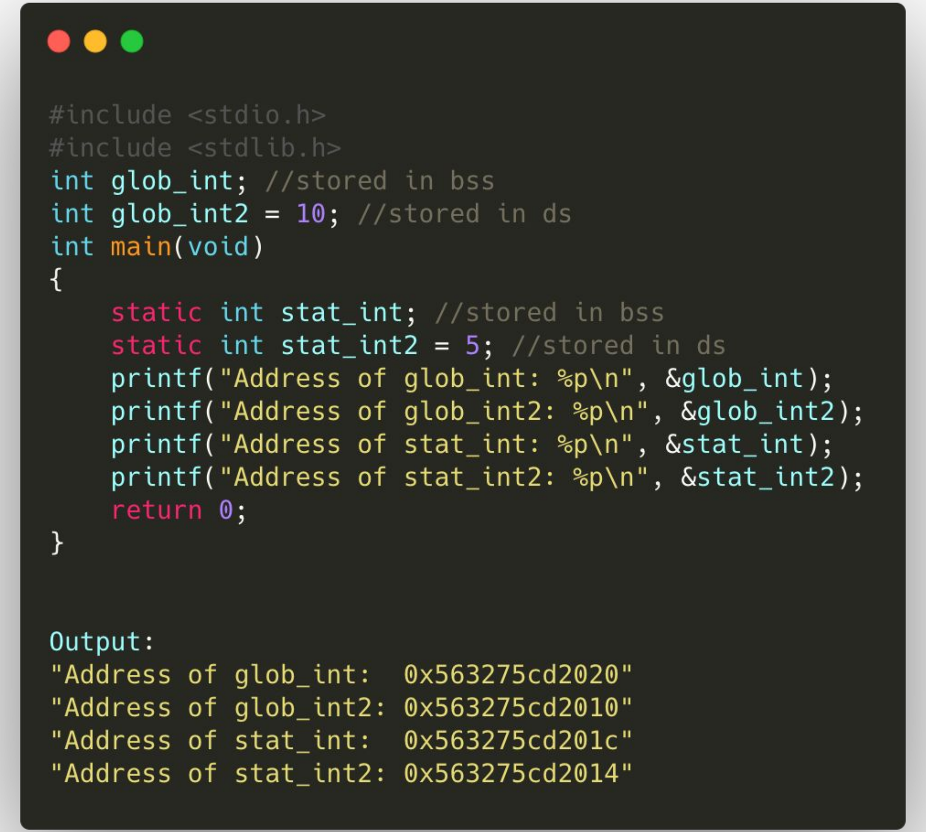
4. Data (DS)



```c
#include <stdio.h>
#include <stdlib.h>
int glob_int;
int glob_int2 = 10;
int main(void)
{
    static int stat_int;
    static int stat_int2 = 5;
    printf("Address of glob_int: %p\n", &glob_int);
    printf("Address of glob_int2: %p\n", &glob_int2);
    printf("Address of stat_int: %p\n", &stat_int);
    printf("Address of stat_int2: %p\n", &stat_int2);
    return 0;
}
```

High	Environment
	Stack ↓
	. . .　　　Empty . .
	Heap ↑
	BSS
	Data(DS)
Low	Text

4. Data (DS)



```
#include <stdio.h>
#include <stdlib.h>
int glob_int; //stored in bss
int glob_int2 = 10; //stored in ds
int main(void)
{
    static int stat_int; //stored in bss
    static int stat_int2 = 5; //stored in ds
    printf("Address of glob_int: %p\n", &glob_int);
    printf("Address of glob_int2: %p\n", &glob_int2);
    printf("Address of stat_int: %p\n", &stat_int);
    printf("Address of stat_int2: %p\n", &stat_int2);
    return 0;
}


Output:
"Address of glob_int:  0x563275cd2020"
"Address of glob_int2: 0x563275cd2010"
"Address of stat_int:  0x563275cd201c"
"Address of stat_int2: 0x563275cd2014"
```

High	Environment
	Stack ↓
	. . . . . Empty
	Heap ↑
	BSS
	Data(DS)
Low	Text

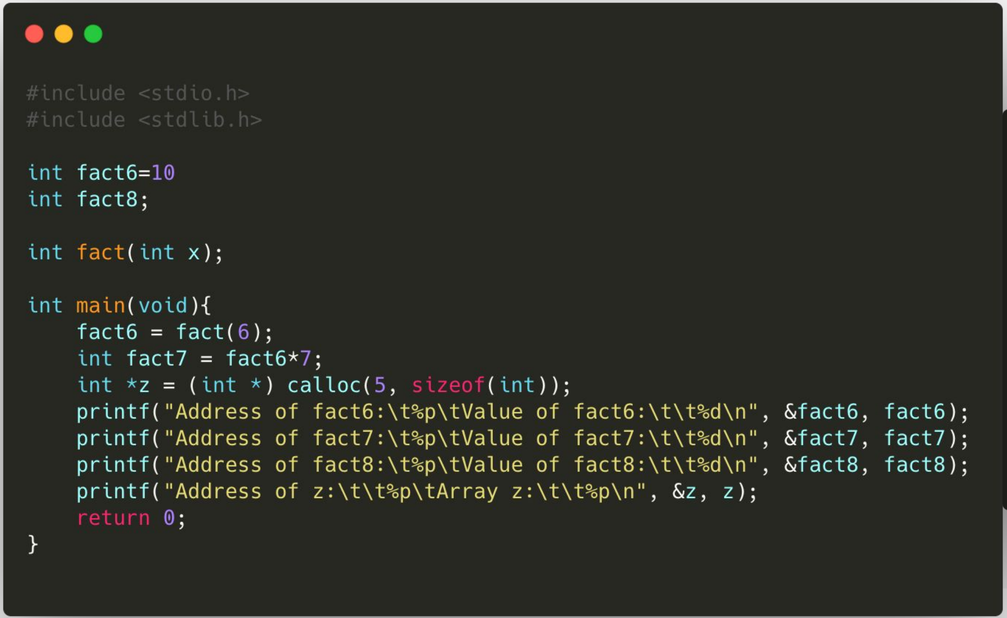
5. Text

- Also called code segment
- The text segment contains executable instructions.
- The text segment is a **read-only segment** that prevents a program from being accidentally modified.

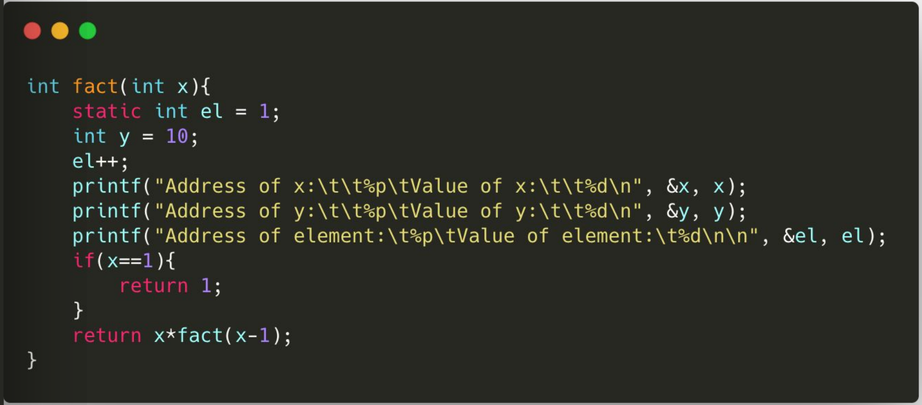
High	Environment
	Stack ↓
	. . . Empty . .
	Heap ↑
	BSS
	Data(DS)
Low	Text

https://aticleworld.com/memory-layout-of-c-program/

Example 1

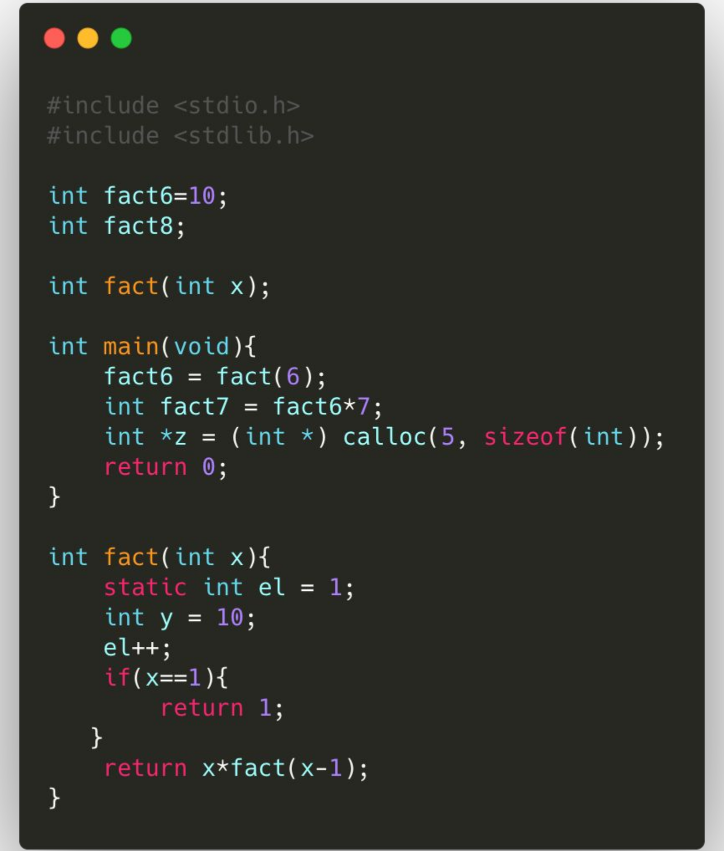


```c
#include <stdio.h>
#include <stdlib.h>

int fact6=10
int fact8;

int fact(int x);

int main(void){
    fact6 = fact(6);
    int fact7 = fact6*7;
    int *z = (int *) calloc(5, sizeof(int));
    printf("Address of fact6:\t%p\tValue of fact6:\t\t%d\n", &fact6, fact6);
    printf("Address of fact7:\t%p\tValue of fact7:\t\t%d\n", &fact7, fact7);
    printf("Address of fact8:\t%p\tValue of fact8:\t\t%d\n", &fact8, fact8);
    printf("Address of z:\t\t%p\tArray z:\t\t%p\n", &z, z);
    return 0;
}
```



```c
int fact(int x){
    static int el = 1;
    int y = 10;
    el++;
    printf("Address of x:\t\t%p\tValue of x:\t\t%d\n", &x, x);
    printf("Address of y:\t\t%p\tValue of y:\t\t%d\n", &y, y);
    printf("Address of element:\t%p\tValue of element:\t%d\n\n", &el, el);
    if(x==1){
        return 1;
    }
    return x*fact(x-1);
}
```

Example 1



```c
#include <stdio.h>
#include <stdlib.h>

int fact6=10;
int fact8;

int fact(int x);

int main(void){
    fact6 = fact(6);
    int fact7 = fact6*7;
    int *z = (int *) calloc(5, sizeof(int));
    return 0;
}

int fact(int x){
    static int el = 1;
    int y = 10;
    el++;
    if(x==1){
        return 1;
   }
    return x*fact(x-1);
}
```

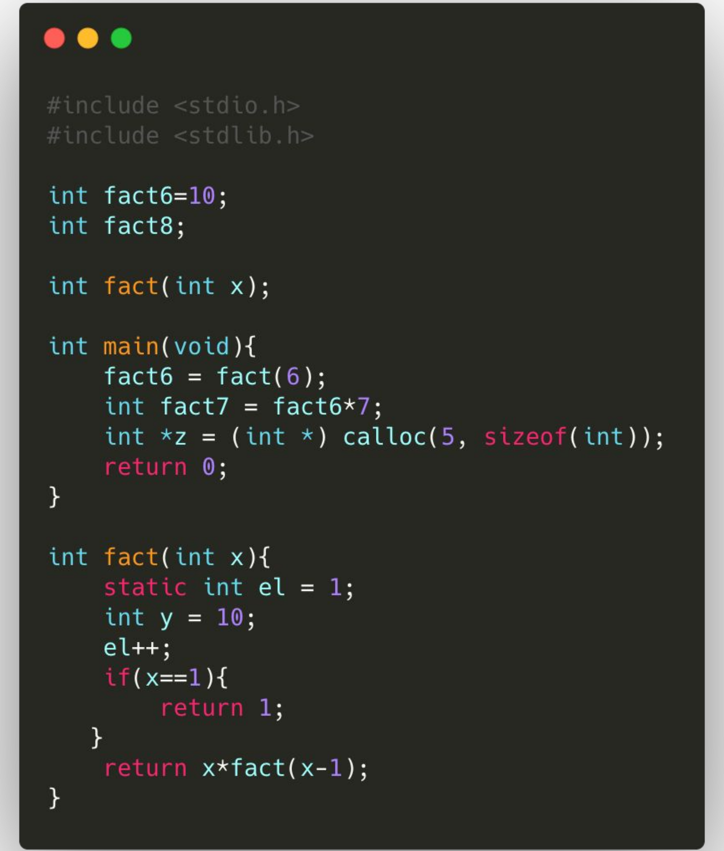
fact6

BSS?

DS?

High	Environment
	Stack ↓
	. . . Empty . .
	Heap ↑
	BSS
	Data(DS)
Low	Text

Example 1

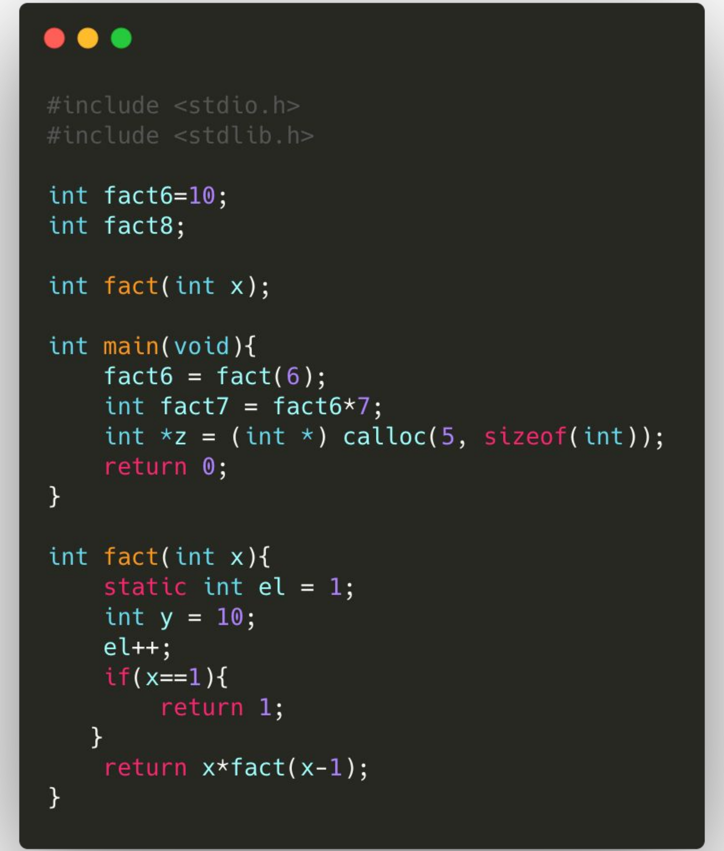


```c
#include <stdio.h>
#include <stdlib.h>

int fact6=10;
int fact8;

int fact(int x);

int main(void){
    fact6 = fact(6);
    int fact7 = fact6*7;
    int *z = (int *) calloc(5, sizeof(int));
    return 0;
}

int fact(int x){
    static int el = 1;
    int y = 10;
    el++;
    if(x==1){
        return 1;
    }
    return x*fact(x-1);
}
```

fact6:
DS

High	Environment
	Stack ↓
	. . . Empty . .
	Heap ↑
	BSS
	Data(DS) fact6
Low	Text

Example 1



```c
#include <stdio.h>
#include <stdlib.h>

int fact6=10;
int fact8;

int fact(int x);

int main(void){
    fact6 = fact(6);
    int fact7 = fact6*7;
    int *z = (int *) calloc(5, sizeof(int));
    return 0;
}

int fact(int x){
    static int el = 1;
    int y = 10;
    el++;
    if(x==1){
        return 1;
    }
    return x*fact(x-1);
}
```

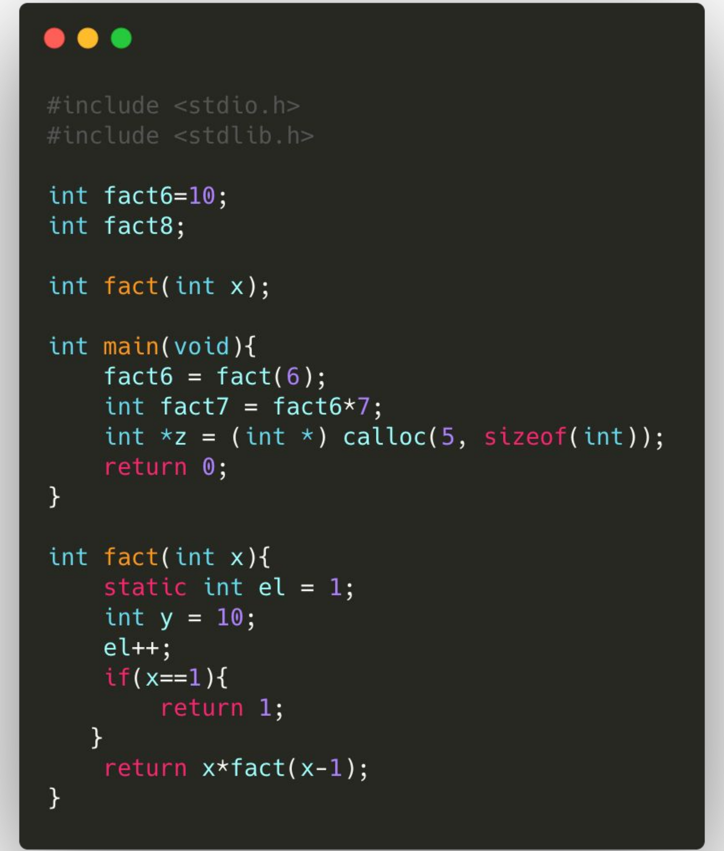
fact8

BSS?

DS?

High	Environment
	Stack ↓
	. . . . . Empty
	Heap ↑
	BSS
	Data(DS) fact6
Low	Text

Example 1

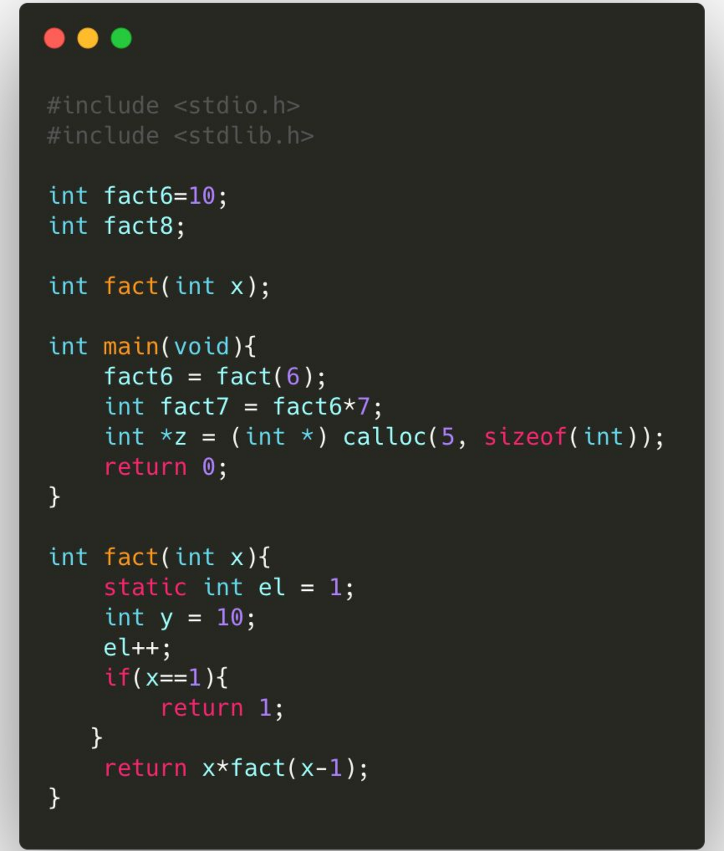


```c
#include <stdio.h>
#include <stdlib.h>

int fact6=10;
int fact8;

int fact(int x);

int main(void){
    fact6 = fact(6);
    int fact7 = fact6*7;
    int *z = (int *) calloc(5, sizeof(int));
    return 0;
}

int fact(int x){
    static int el = 1;
    int y = 10;
    el++;
    if(x==1){
        return 1;
   }
    return x*fact(x-1);
}
```

fact8:
BSS

High	Environment
	Stack ↓
	. . . Empty . .
	Heap ↑
	BSS fact8
	Data(DS) fact6
Low	Text

Example 1



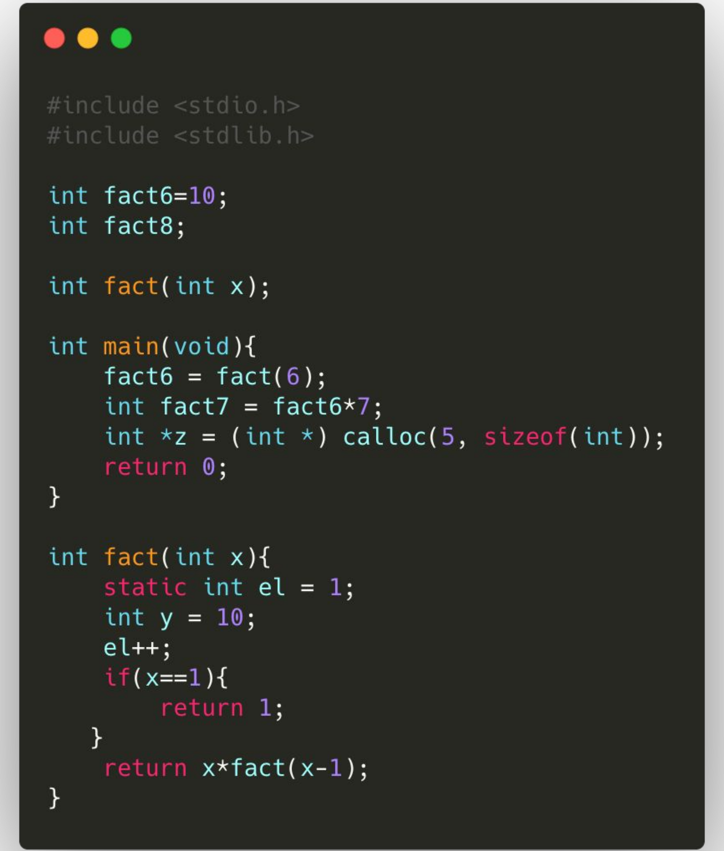
```c
#include <stdio.h>
#include <stdlib.h>

int fact6=10;
int fact8;

int fact(int x);

int main(void){
    fact6 = fact(6);
    int fact7 = fact6*7;
    int *z = (int *) calloc(5, sizeof(int));
    return 0;
}

int fact(int x){
    static int el = 1;
    int y = 10;
    el++;
    if(x==1){
        return 1;
   }
    return x*fact(x-1);
}
```

fact7

Stack?
Heap?

High	Environment
	Stack ↓
	. . . . . Empty
	Heap ↑
	BSS fact8
	Data(DS) fact6
Low	Text

Example 1

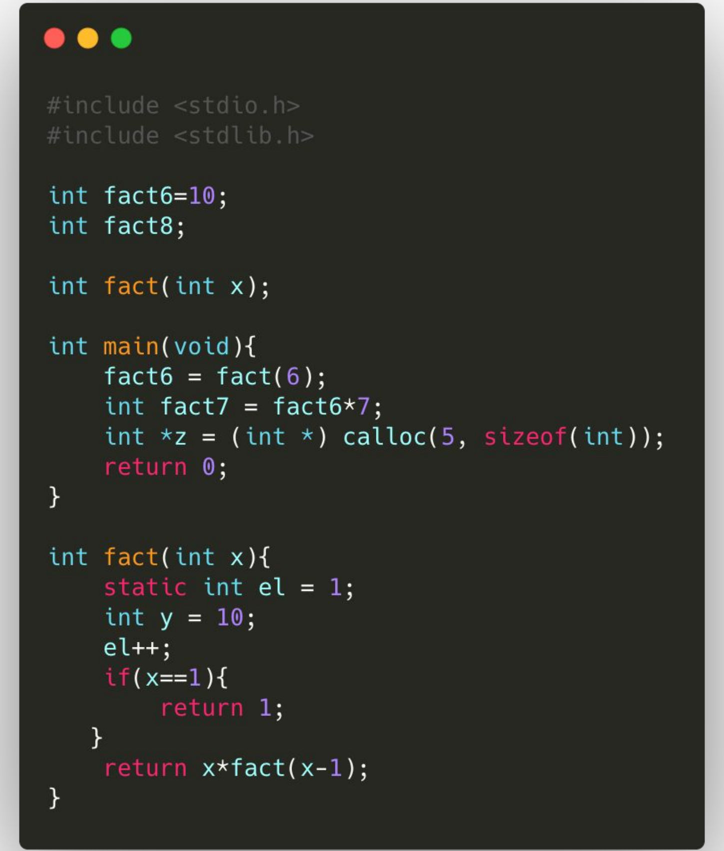


```c
#include <stdio.h>
#include <stdlib.h>

int fact6=10;
int fact8;

int fact(int x);

int main(void){
    fact6 = fact(6);
    int fact7 = fact6*7;
    int *z = (int *) calloc(5, sizeof(int));
    return 0;
}

int fact(int x){
    static int el = 1;
    int y = 10;
    el++;
    if(x==1){
        return 1;
   }
    return x*fact(x-1);
}
```

fact7:
Stack

High	Environment
	Stack ↓
	fact7 . . Empty . .
	Heap ↑
	BSS fact8
	Data(DS) fact6
Low	Text

Example 1



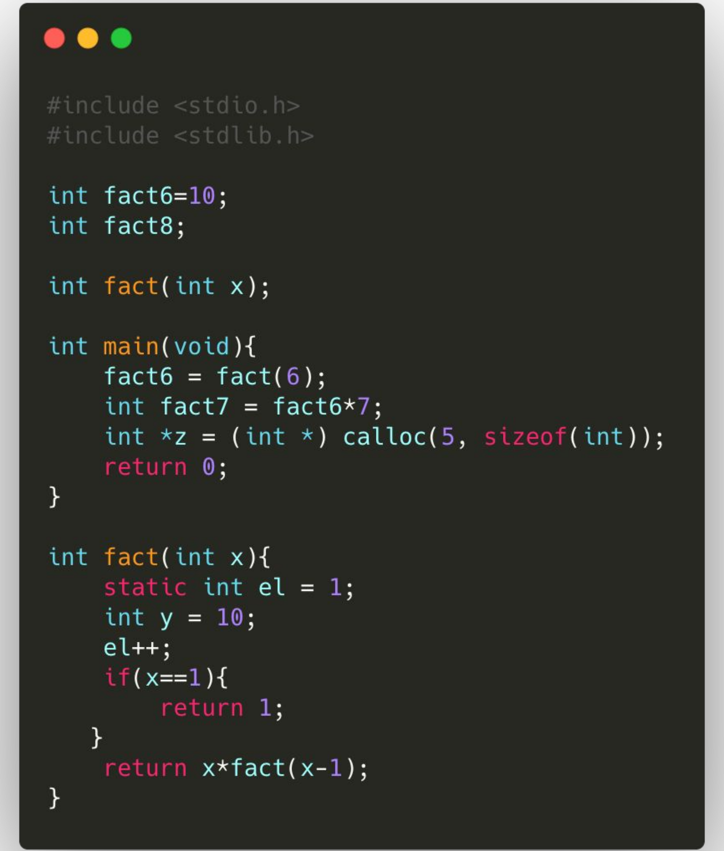
```c
#include <stdio.h>
#include <stdlib.h>

int fact6=10;
int fact8;

int fact(int x);

int main(void){
    fact6 = fact(6);
    int fact7 = fact6*7;
    int *z = (int *) calloc(5, sizeof(int));
    return 0;
}

int fact(int x){
    static int el = 1;
    int y = 10;
    el++;
    if(x==1){
        return 1;
    }
    return x*fact(x-1);
}
```

z

Stack?
Heap?

High	Environment
	Stack ↓
	fact7 . . Empty . .
	Heap ↑
	BSS fact8
	Data(DS) fact6
Low	Text

Example 1



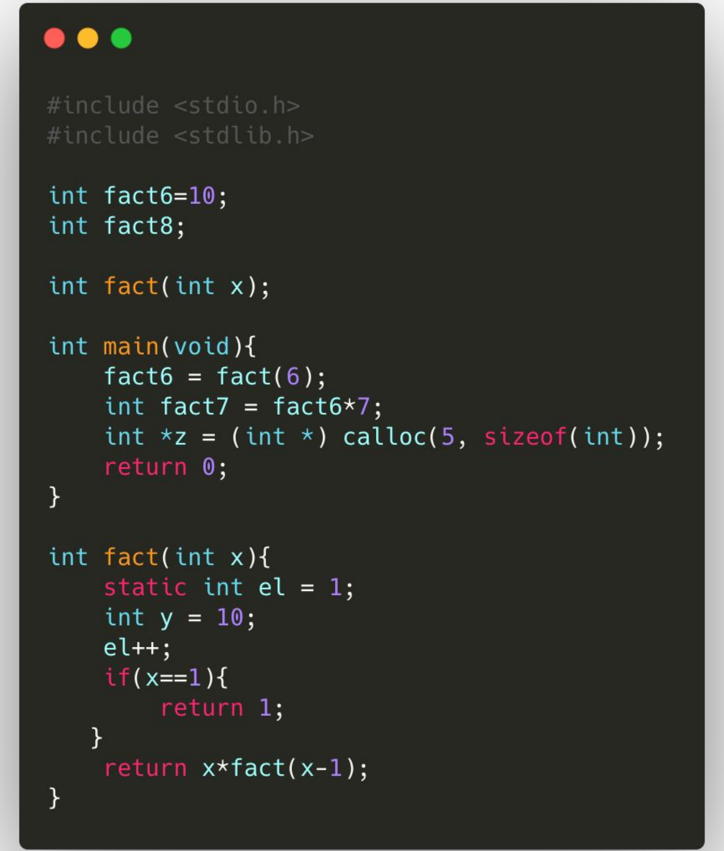
```c
#include <stdio.h>
#include <stdlib.h>

int fact6=10;
int fact8;

int fact(int x);

int main(void){
    fact6 = fact(6);
    int fact7 = fact6*7;
    int *z = (int *) calloc(5, sizeof(int));
    return 0;
}

int fact(int x){
    static int el = 1;
    int y = 10;
    el++;
    if(x==1){
        return 1;
   }
    return x*fact(x-1);
}
```

z

Stack

High	Environment
	Stack ↓
	fact7 z . Empty . .
	Heap ↑
	BSS fact8
	Data(DS) fact6
Low	Text

Example 1



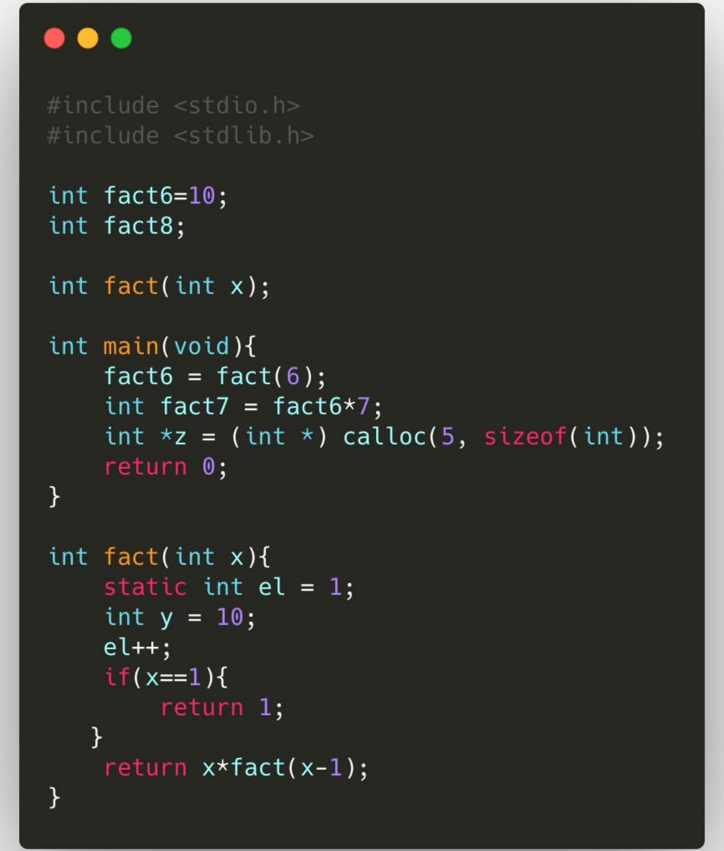
```c
#include <stdio.h>
#include <stdlib.h>

int fact6=10;
int fact8;

int fact(int x);

int main(void){
    fact6 = fact(6);
    int fact7 = fact6*7;
    int *z = (int *) calloc(5, sizeof(int));
    return 0;
}

int fact(int x){
    static int el = 1;
    int y = 10;
    el++;
    if(x==1){
        return 1;
    }
    return x*fact(x-1);
}
```

Address that z points

Stack?
Heap?

High	Environment
	Stack ↓
	fact7 z . . . Empty
	Heap ↑
	BSS fact8
	Data(DS) fact6
Low	Text

Example 1

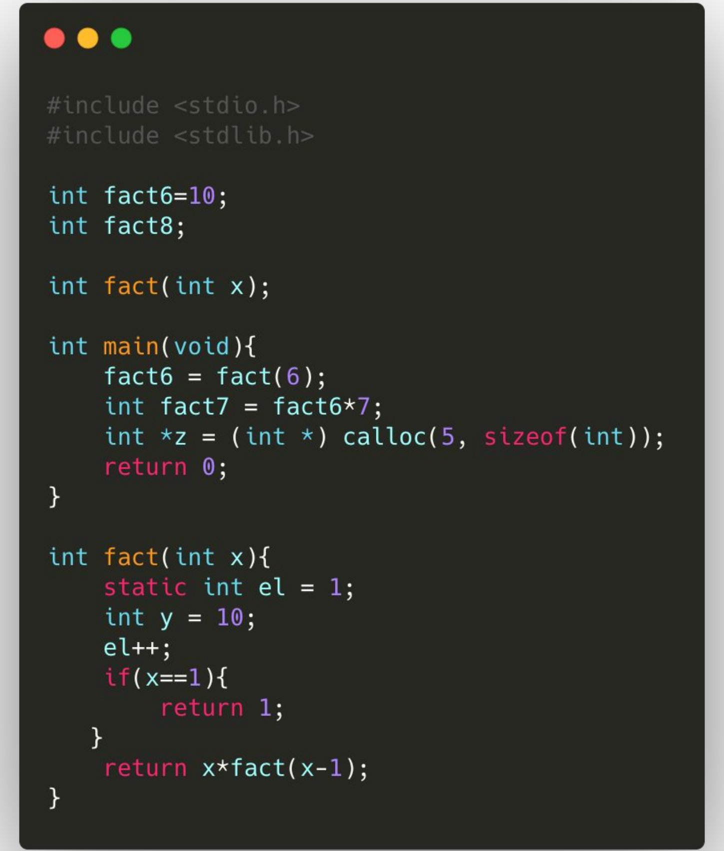


```c
#include <stdio.h>
#include <stdlib.h>

int fact6=10;
int fact8;

int fact(int x);

int main(void){
    fact6 = fact(6);
    int fact7 = fact6*7;
    int *z = (int *) calloc(5, sizeof(int));
    return 0;
}

int fact(int x){
    static int el = 1;
    int y = 10;
    el++;
    if(x==1){
        return 1;
    }
    return x*fact(x-1);
}
```

Address that z points: Heap

High	Environment
	Stack ↓
	fact7 z . . . Empty
	Heap ↑
	BSS fact8
	Data(DS) fact6
Low	Text

Example 1

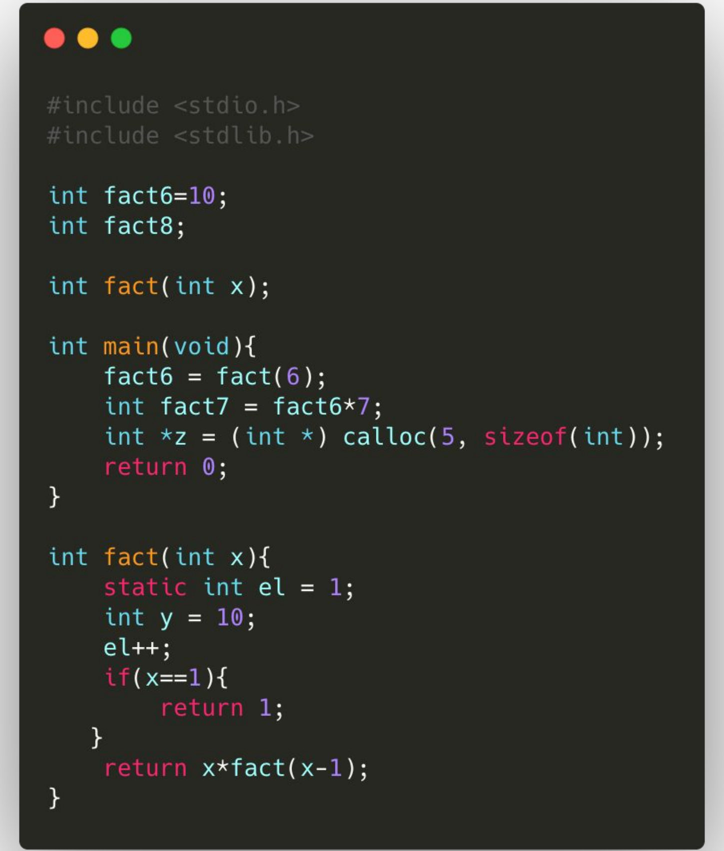


```c
#include <stdio.h>
#include <stdlib.h>

int fact6=10;
int fact8;

int fact(int x);

int main(void){
    fact6 = fact(6);
    int fact7 = fact6*7;
    int *z = (int *) calloc(5, sizeof(int));
    return 0;
}

int fact(int x){
    static int el = 1;
    int y = 10;
    el++;
    if(x==1){
        return 1;
    }
    return x*fact(x-1);
}
```

Variables in:
fact(6)
fact(5)
fact(4)
fact(3)
fact(2)
fact(1)

High	Environment
	Stack ↓
	fact7 z . . Empty .
	Heap ↑
	BSS fact8
	Data(DS) fact6
Low	Text

Example 1



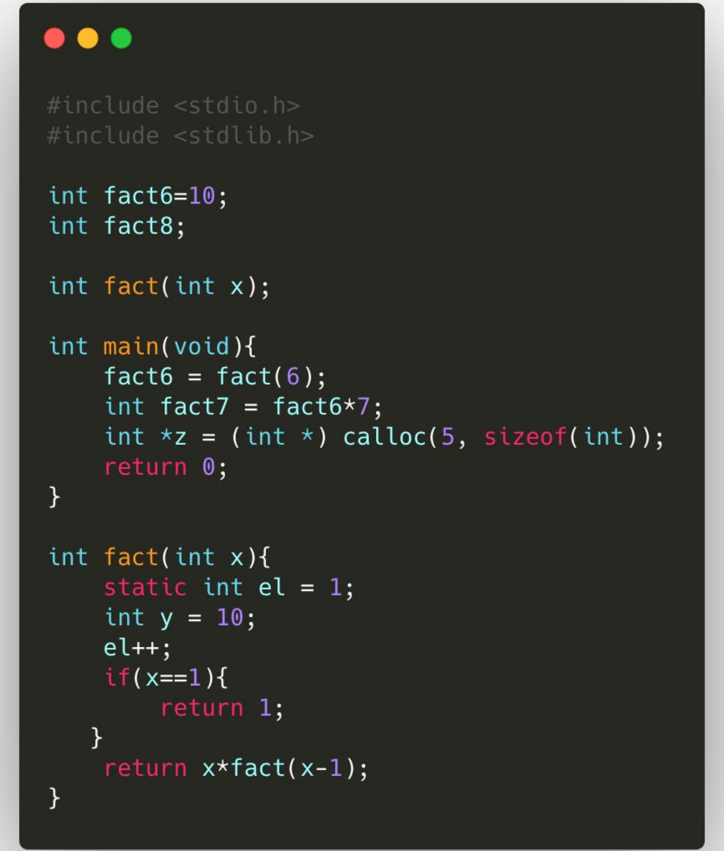
```c
#include <stdio.h>
#include <stdlib.h>

int fact6=10;
int fact8;

int fact(int x);

int main(void){
    fact6 = fact(6);
    int fact7 = fact6*7;
    int *z = (int *) calloc(5, sizeof(int));
    return 0;
}

int fact(int x){
    static int el = 1;
    int y = 10;
    el++;
    if(x==1){
        return 1;
    }
    return x*fact(x-1);
}
```

Variables in:
fact(6)
fact(5)
fact(4)
fact(3)
fact(2)
fact(1)

Each one will be in separate stack frames! Be careful about static variables.

High	Environment
	Stack ↓
	fact7 z . . Empty .
	Heap ↑
	BSS fact8
	Data(DS) fact6
Low	Text

Example 1



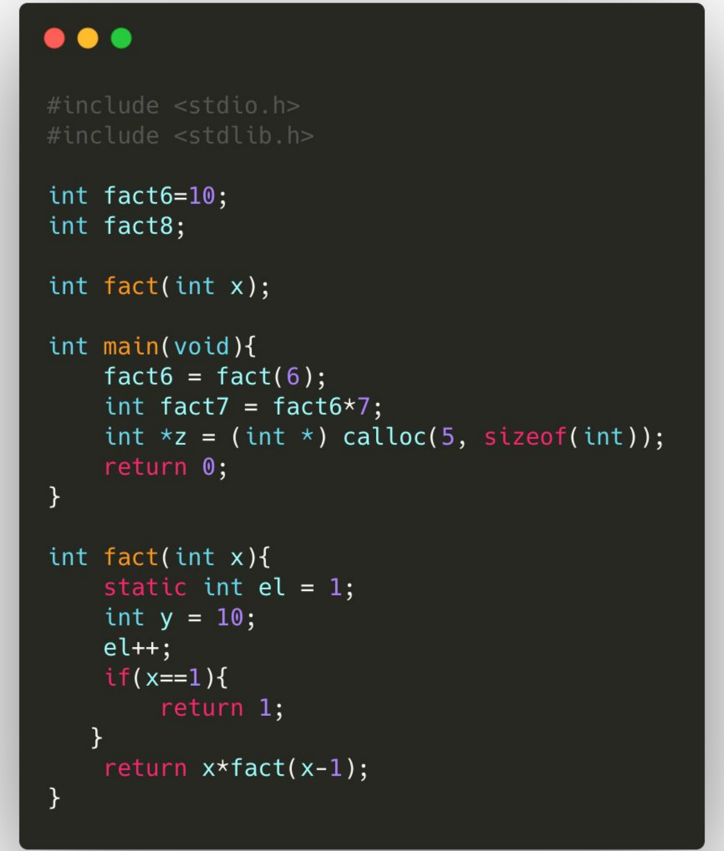
```c
#include <stdio.h>
#include <stdlib.h>

int fact6=10;
int fact8;

int fact(int x);

int main(void){
    fact6 = fact(6);
    int fact7 = fact6*7;
    int *z = (int *) calloc(5, sizeof(int));
    return 0;
}

int fact(int x){
    static int el = 1;
    int y = 10;
    el++;
    if(x==1){
        return 1;
    }
    return x*fact(x-1);
}
```

Variables in:
fact(6)
el?

Stack?
DS?

High	Environment
	Stack ↓
	fact7 z . . Empty .
	Heap ↑
	BSS fact8
	Data(DS) fact6
Low	Text

Example 1

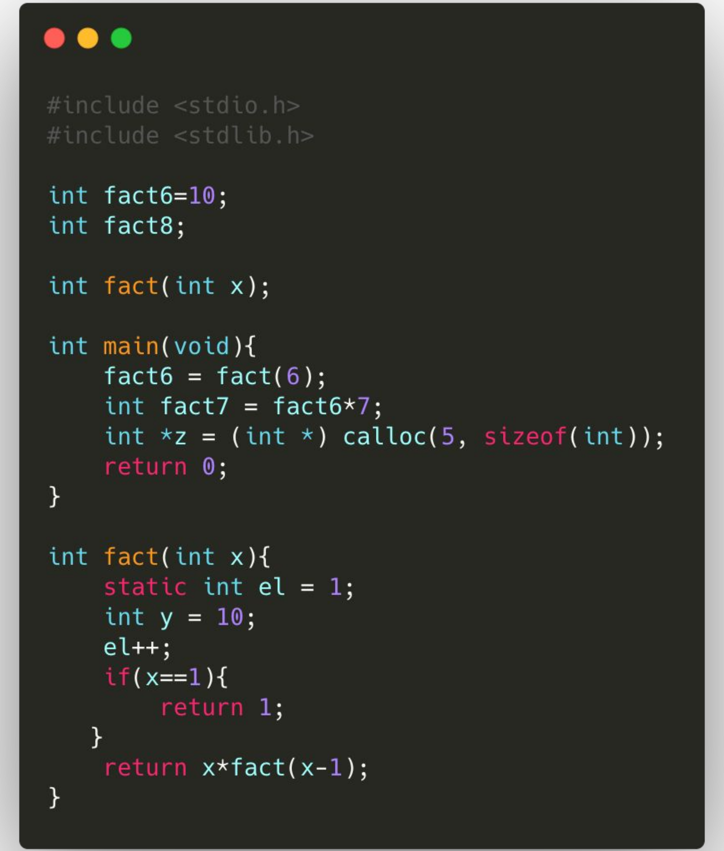


```c
#include <stdio.h>
#include <stdlib.h>

int fact6=10;
int fact8;

int fact(int x);

int main(void){
    fact6 = fact(6);
    int fact7 = fact6*7;
    int *z = (int *) calloc(5, sizeof(int));
    return 0;
}

int fact(int x){
    static int el = 1;
    int y = 10;
    el++;
    if(x==1){
        return 1;
   }
    return x*fact(x-1);
}
```

Variables in:
fact(6)
el: DS

High	Environment
	Stack ↓
	fact7 . . Empty .
	Heap ↑
	BSS fact8
	Data(DS) fact6 el
Low	Text

Example 1



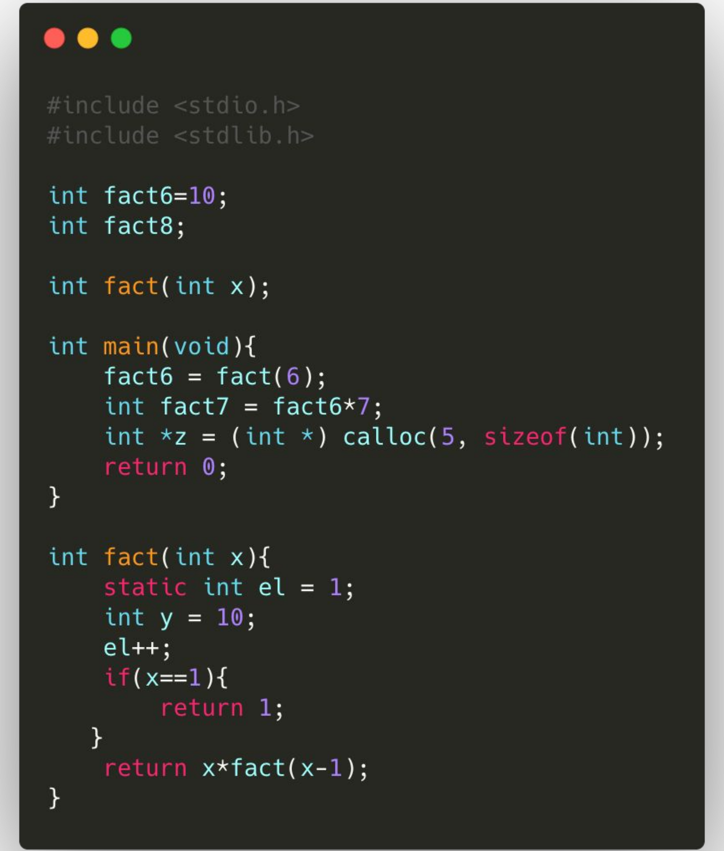
```c
#include <stdio.h>
#include <stdlib.h>

int fact6=10;
int fact8;

int fact(int x);

int main(void){
    fact6 = fact(6);
    int fact7 = fact6*7;
    int *z = (int *) calloc(5, sizeof(int));
    return 0;
}

int fact(int x){
    static int el = 1;
    int y = 10;
    el++;
    if(x==1){
        return 1;
    }
    return x*fact(x-1);
}
```

Variables in:
fact(6)
el: DS
y, x : ?

DS?
Stack?

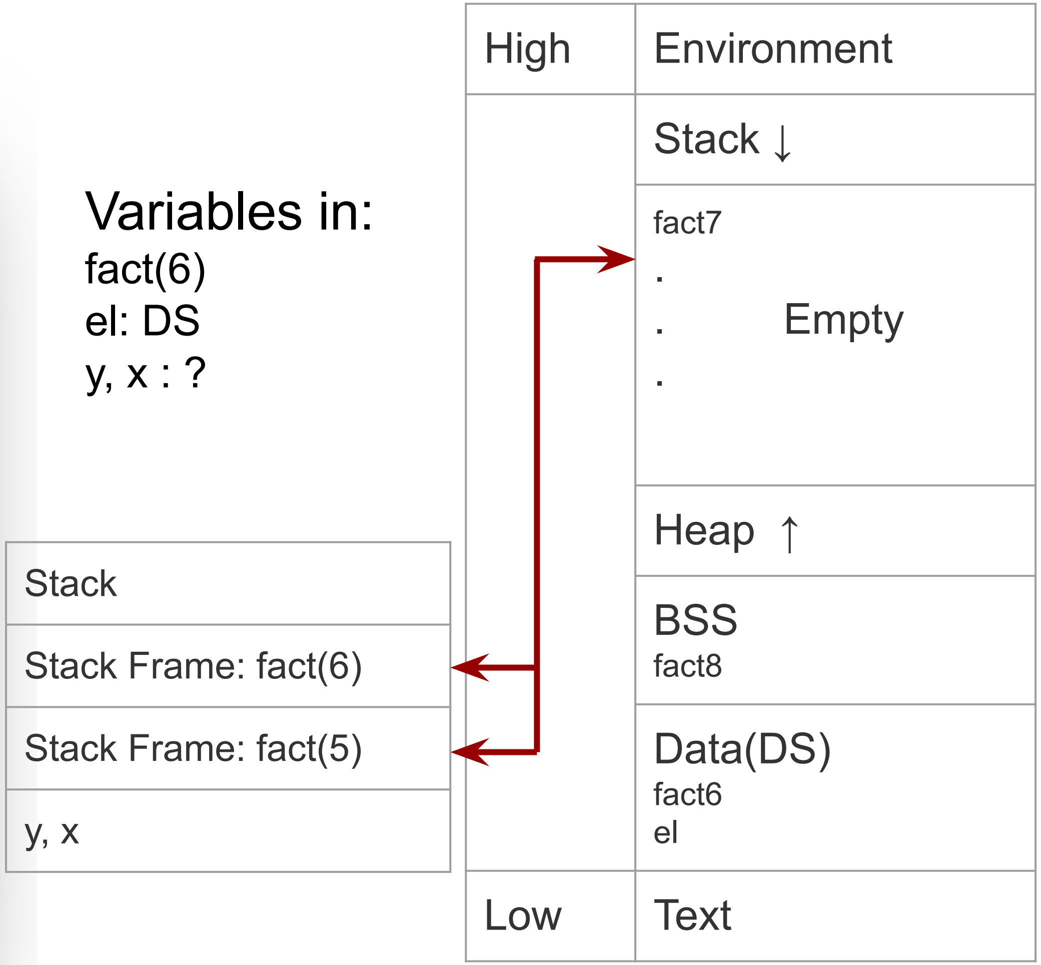
High	Environment
	Stack ↓
	fact7 z . . Empty .
	Heap ↑
	BSS fact8
	Data(DS) fact6 el
Low	Text

Example 1



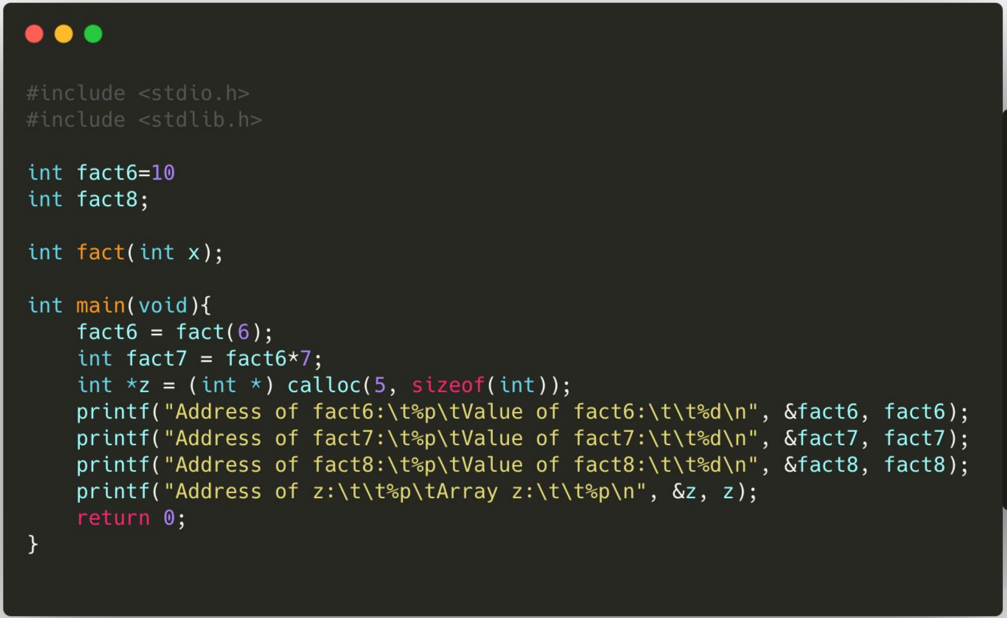
```c
#include <stdio.h>
#include <stdlib.h>

int fact6=10;
int fact8;

int fact(int x);

int main(void){
    fact6 = fact(6);
    int fact7 = fact6*7;
    int *z = (int *) calloc(5, sizeof(int));
    return 0;
}

int fact(int x){
    static int el = 1;
    int y = 10;
    el++;
    if(x==1){
        return 1;
    }
    return x*fact(x-1);
}
```

Variables in:
fact(6)
el: DS
y, x : ?

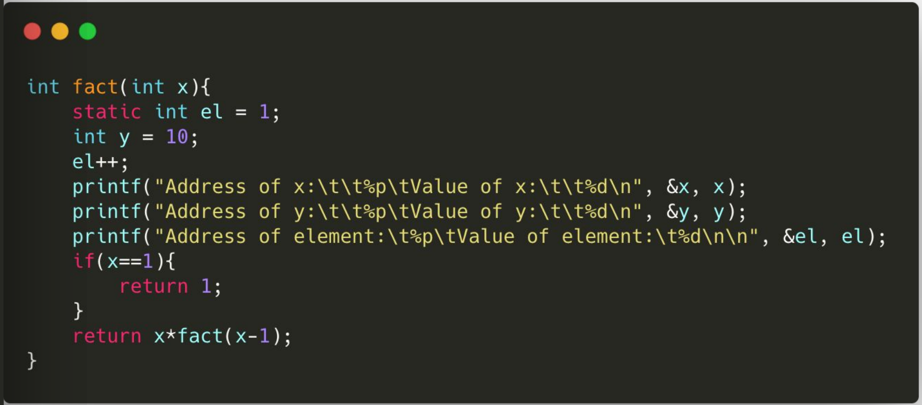


Example 1

Let’s look at the address of variables:

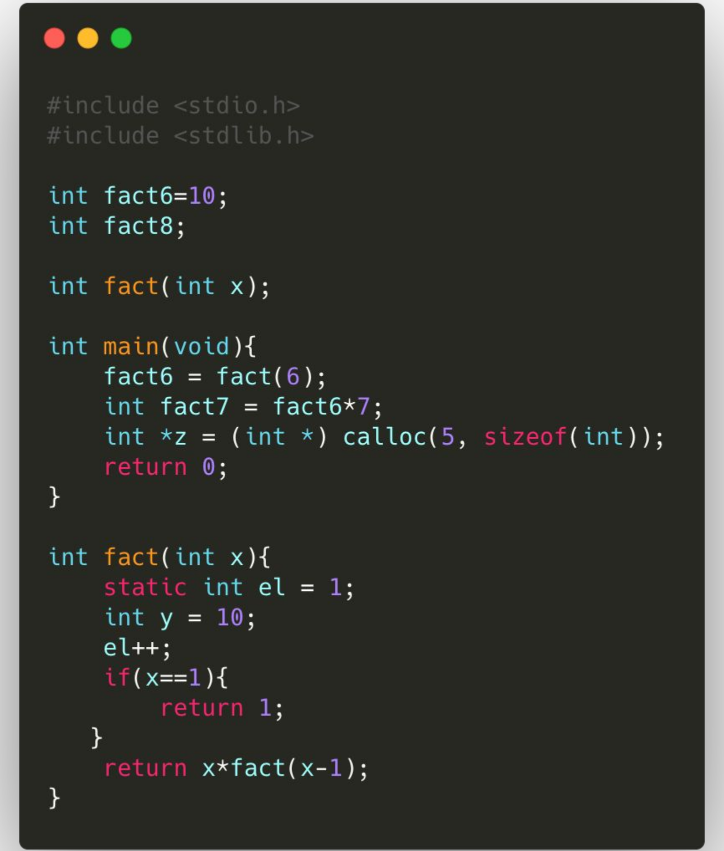


```c
#include <stdio.h>
#include <stdlib.h>

int fact6=10
int fact8;

int fact(int x);

int main(void){
    fact6 = fact(6);
    int fact7 = fact6*7;
    int *z = (int *) calloc(5, sizeof(int));
    printf("Address of fact6:\t%p\tValue of fact6:\t\t%d\n", &fact6, fact6);
    printf("Address of fact7:\t%p\tValue of fact7:\t\t%d\n", &fact7, fact7);
    printf("Address of fact8:\t%p\tValue of fact8:\t\t%d\n", &fact8, fact8);
    printf("Address of z:\t\t%p\tArray z:\t\t%p\n", &z, z);
    return 0;
}
```



```c
int fact(int x){
    static int el = 1;
    int y = 10;
    el++;
    printf("Address of x:\t\t%p\tValue of x:\t\t%d\n", &x, x);
    printf("Address of y:\t\t%p\tValue of y:\t\t%d\n", &y, y);
    printf("Address of element:\t%p\tValue of element:\t%d\n\n", &el, el);
    if(x==1){
        return 1;
    }
    return x*fact(x-1);
}
```

Example 1



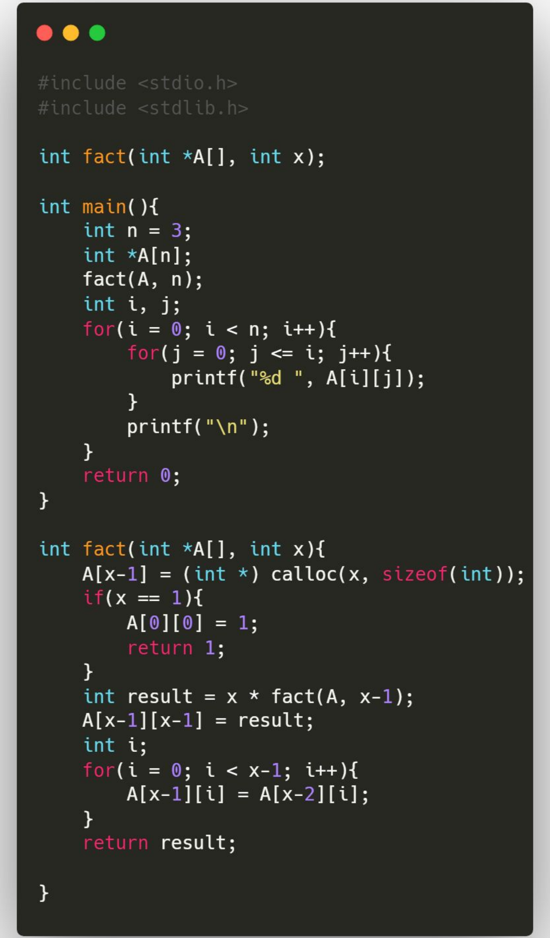
```c
#include <stdio.h>
#include <stdlib.h>

int fact6=10;
int fact8;

int fact(int x);

int main(void){
    fact6 = fact(6);
    int fact7 = fact6*7;
    int *z = (int *) calloc(5, sizeof(int));
    return 0;
}

int fact(int x){
    static int el = 1;
    int y = 10;
    el++;
    if(x==1){
        return 1;
    }
    return x*fact(x-1);
}
```

Stack	Address	Variable	Value
	0x7fff53810750	&z	0x55e9f90d3670
	0x7fff5381074c	fact7	5040
Stack Frame for fact(6)	0x7fff53810724	y in fact(6)	10
	0x7fff5381071c	x in fact(6)	6
Stack Frame for fact(5)	0x7fff538106f4	y in fact(5)	10
	0x7fff538106ec	x in fact(5)	5
Stack Frame for fact(4)	0x7fff538106c4	y in fact(4)	10
	0x7fff538106bc	x in fact(4)	4
Stack Frame for fact(3)	0x7fff53810694	y in fact(3)	10
	0x7fff5381068c	x in fact(3)	3
Stack Frame for fact(2)	0x7fff53810664	y in fact(2)	10
	0x7fff5381065c	x in fact(2)	2
Stack Frame for fact(1)	0x7fff53810634	y in fact(1)	10
	0x7fff5381062c	x in fact(1)	1
Heap	0x55e9f90d3670	→z	calloc
BSS	0x55e9f728e01c	fact8	0
Data(DS)	0x55e9f728e014	element	7
	0x55e9f728e010	fact6	720
Text			

Example 2

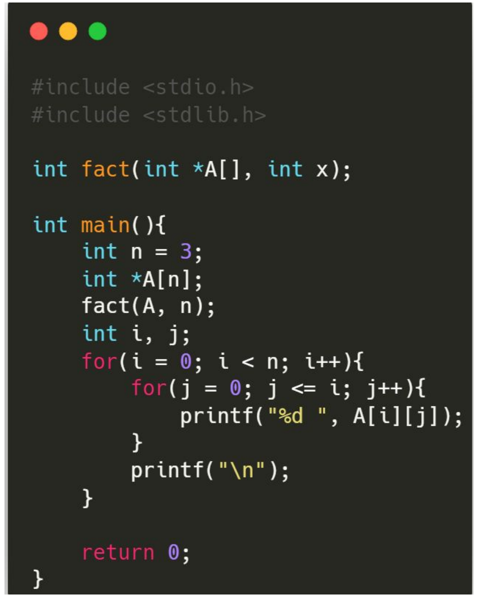
- Computes factorials up to n!
- Outputs

```
1

1  2

1  2  6
```

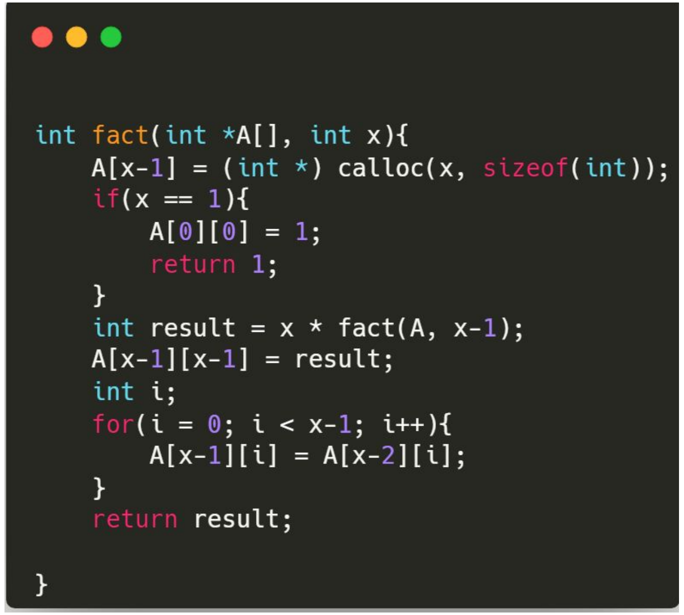


```c
#include <stdio.h>
#include <stdlib.h>

int fact(int *A[], int x);

int main(){
    int n = 3;
    int *A[n];
    fact(A, n);
    int i, j;
    for(i = 0; i < n; i++){
        for(j = 0; j <= i; j++){
            printf("%d ", A[i][j]);
        }
        printf("\n");
    }
    return 0;
}

int fact(int *A[], int x){
    A[x-1] = (int *) calloc(x, sizeof(int));
    if(x == 1){
        A[0][0] = 1;
        return 1;
    }
    int result = x * fact(A, x-1);
    A[x-1][x-1] = result;
    int i;
    for(i = 0; i < x-1; i++){
        A[x-1][i] = A[x-2][i];
    }
    return result;

}
```

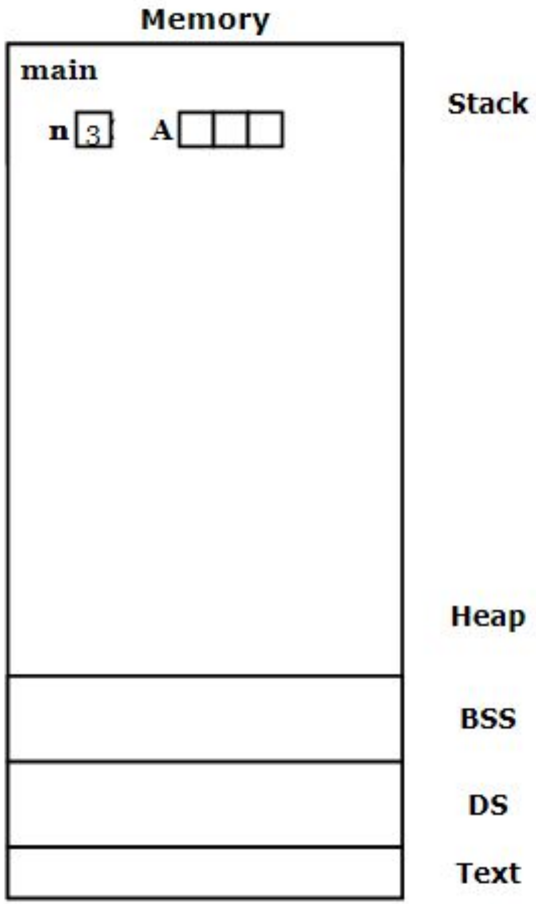
Example 2



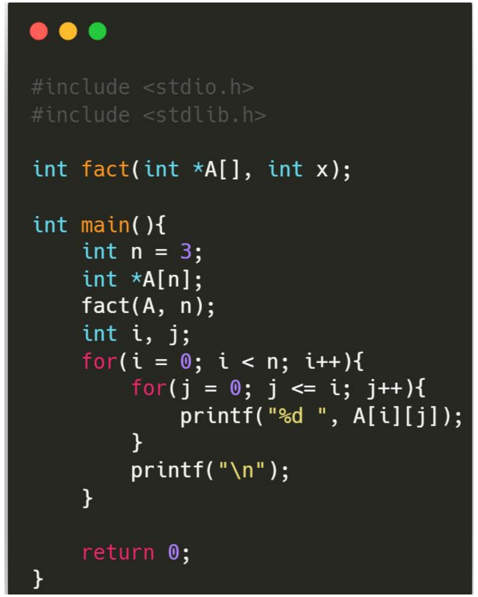
```c
#include <stdio.h>
#include <stdlib.h>

int fact(int *A[], int x);

int main(){
    int n = 3;
    int *A[n];
    fact(A, n);
    int i, j;
    for(i = 0; i < n; i++){
        for(j = 0; j <= i; j++){
            printf("%d ", A[i][j]);
        }
        printf("\n");
    }

    return 0;
}
```



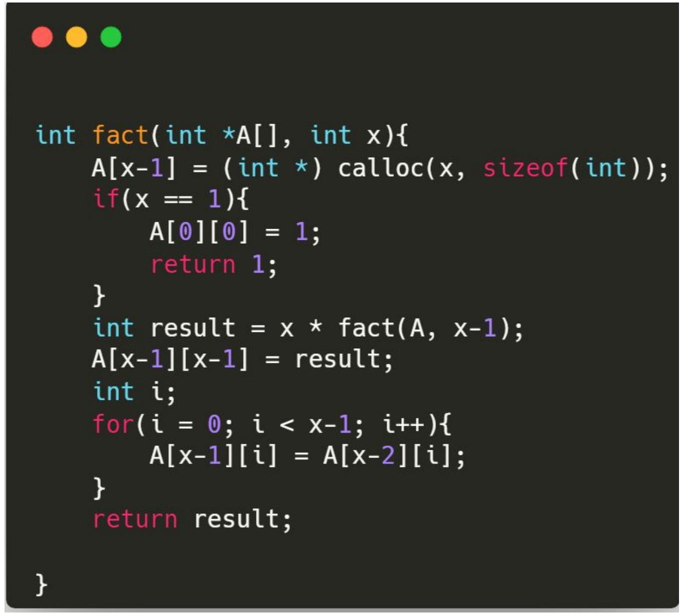
```c
int fact(int *A[], int x){
    A[x-1] = (int *) calloc(x, sizeof(int));
    if(x == 1){
        A[0][0] = 1;
        return 1;
    }
    int result = x * fact(A, x-1);
    A[x-1][x-1] = result;
    int i;
    for(i = 0; i < x-1; i++){
        A[x-1][i] = A[x-2][i];
    }
    return result;

}
```



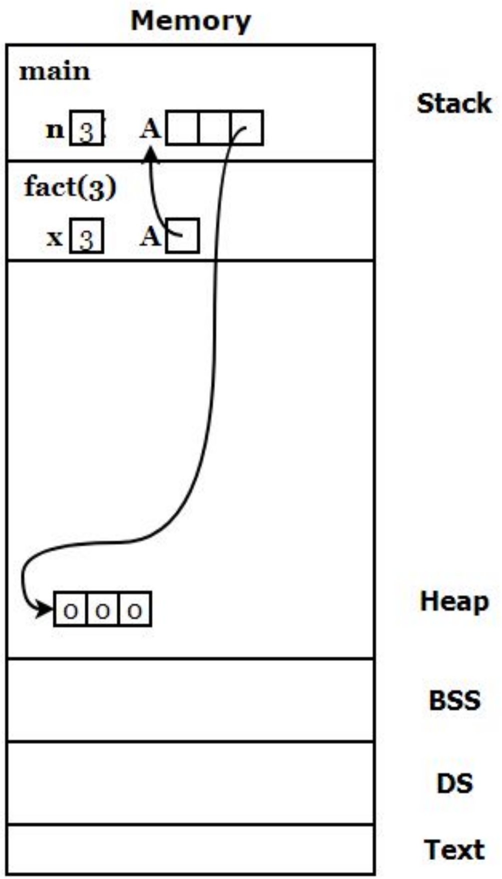
Example 2



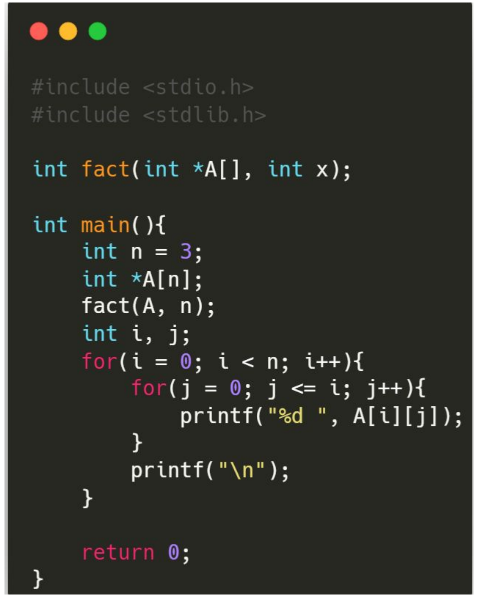
```c
#include <stdio.h>
#include <stdlib.h>

int fact(int *A[], int x);

int main(){
    int n = 3;
    int *A[n];
    fact(A, n);
    int i, j;
    for(i = 0; i < n; i++){
        for(j = 0; j <= i; j++){
            printf("%d ", A[i][j]);
        }
        printf("\n");
    }

    return 0;
}
```



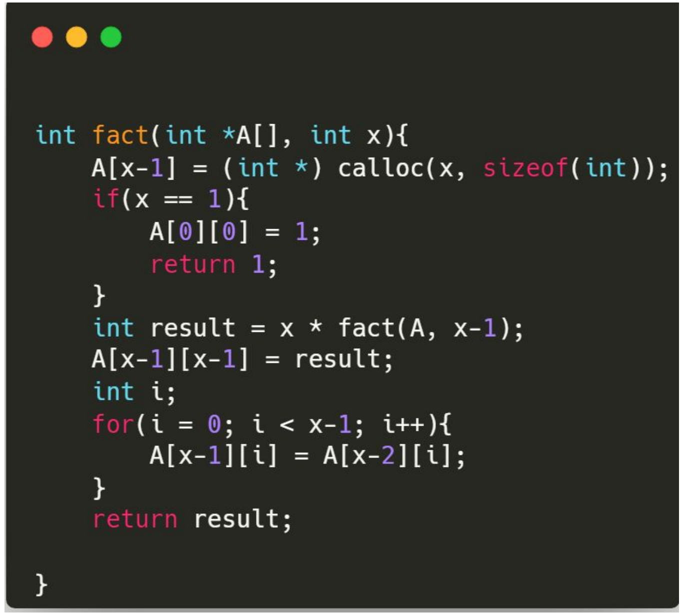
```c
int fact(int *A[], int x){
    A[x-1] = (int *) calloc(x, sizeof(int));
    if(x == 1){
        A[0][0] = 1;
        return 1;
    }
    int result = x * fact(A, x-1);
    A[x-1][x-1] = result;
    int i;
    for(i = 0; i < x-1; i++){
        A[x-1][i] = A[x-2][i];
    }
    return result;

}
```



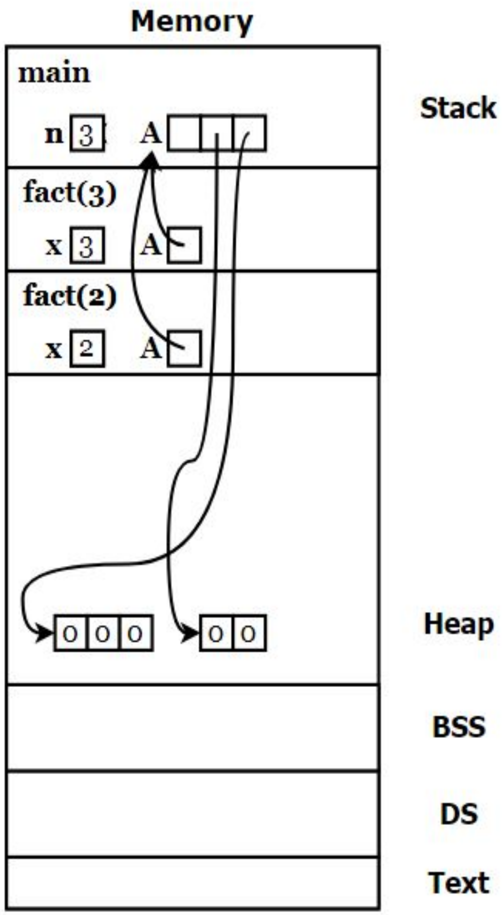
Example 2



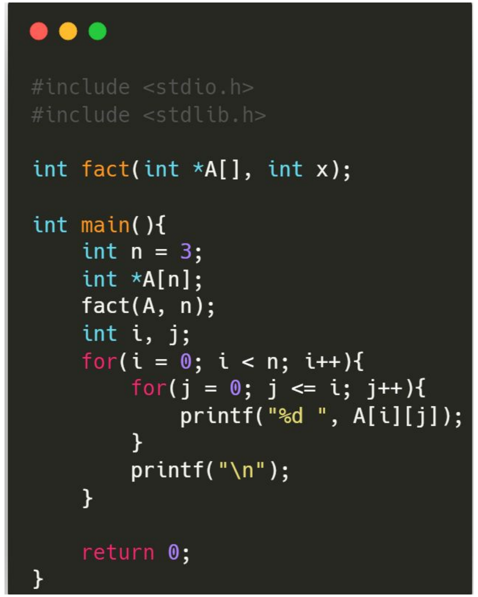
```
#include <stdio.h>
#include <stdlib.h>

int fact(int *A[], int x);

int main(){
    int n = 3;
    int *A[n];
    fact(A, n);
    int i, j;
    for(i = 0; i < n; i++){
        for(j = 0; j <= i; j++){
            printf("%d ", A[i][j]);
        }
        printf("\n");
    }

    return 0;
}
```



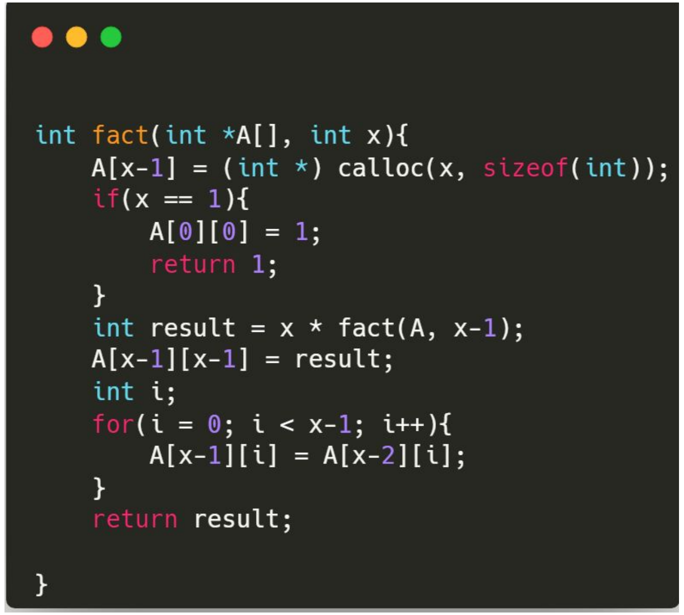
```
int fact(int *A[], int x){
    A[x-1] = (int *) calloc(x, sizeof(int));
    if(x == 1){
        A[0][0] = 1;
        return 1;
    }
    int result = x * fact(A, x-1);
    A[x-1][x-1] = result;
    int i;
    for(i = 0; i < x-1; i++){
        A[x-1][i] = A[x-2][i];
    }
    return result;

}
```



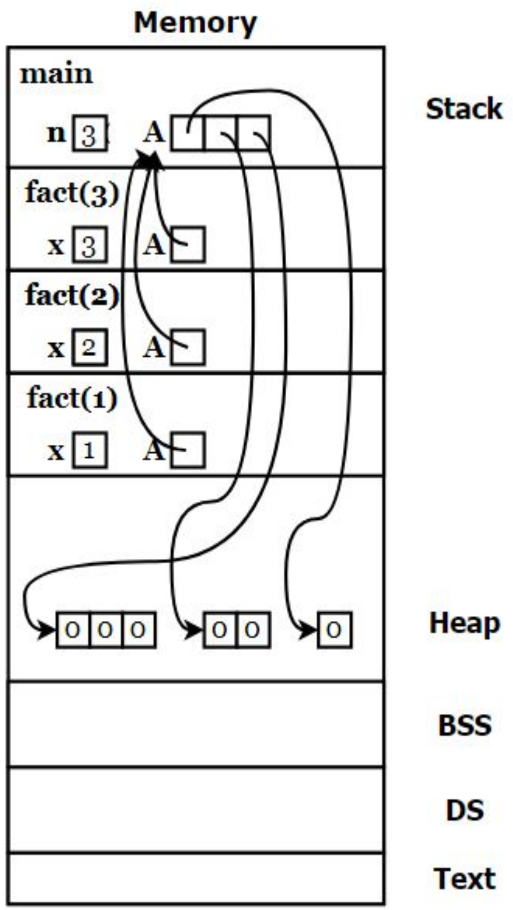
Example 2



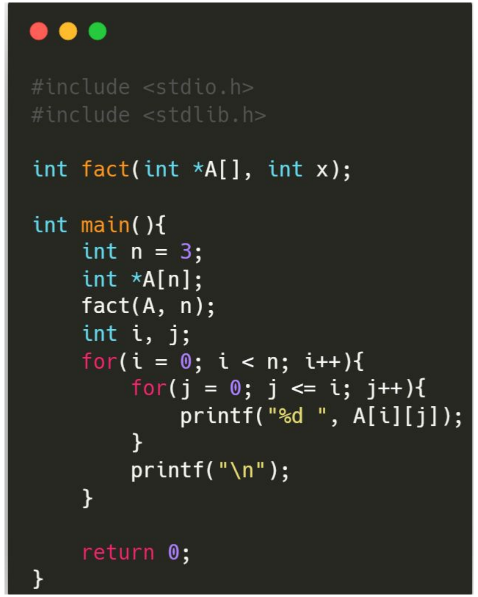
```c
#include <stdio.h>
#include <stdlib.h>

int fact(int *A[], int x);

int main(){
    int n = 3;
    int *A[n];
    fact(A, n);
    int i, j;
    for(i = 0; i < n; i++){
        for(j = 0; j <= i; j++){
            printf("%d ", A[i][j]);
        }
        printf("\n");
    }

    return 0;
}
```



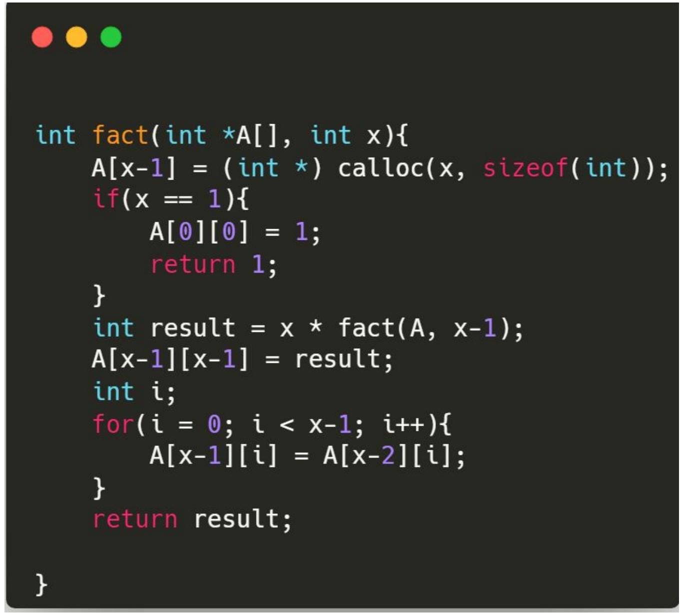
```c
int fact(int *A[], int x){
    A[x-1] = (int *) calloc(x, sizeof(int));
    if(x == 1){
        A[0][0] = 1;
        return 1;
    }
    int result = x * fact(A, x-1);
    A[x-1][x-1] = result;
    int i;
    for(i = 0; i < x-1; i++){
        A[x-1][i] = A[x-2][i];
    }
    return result;

}
```



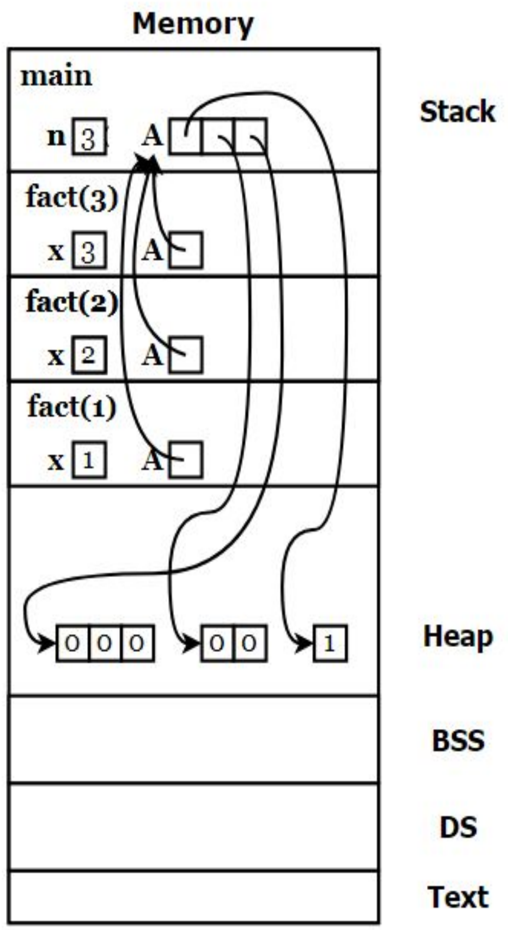
Example 2



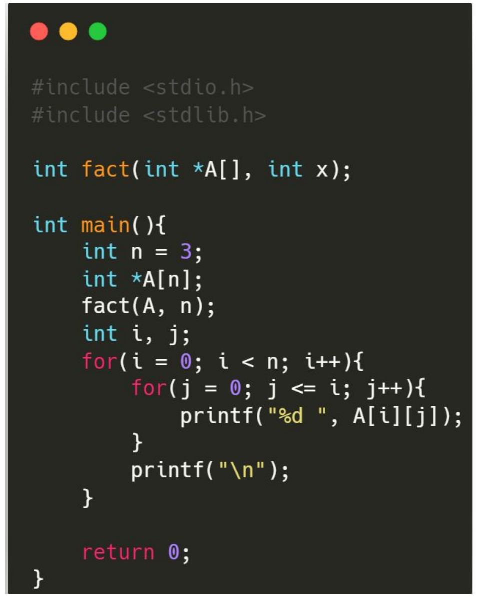
```c
#include <stdio.h>
#include <stdlib.h>

int fact(int *A[], int x);

int main(){
    int n = 3;
    int *A[n];
    fact(A, n);
    int i, j;
    for(i = 0; i < n; i++){
        for(j = 0; j <= i; j++){
            printf("%d ", A[i][j]);
        }
        printf("\n");
    }

    return 0;
}
```



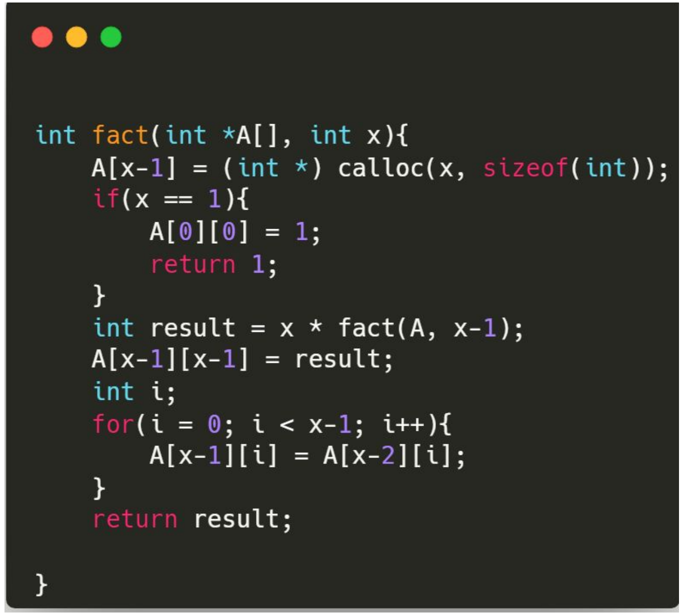
```c
int fact(int *A[], int x){
    A[x-1] = (int *) calloc(x, sizeof(int));
    if(x == 1){
        A[0][0] = 1;
        return 1;
    }
    int result = x * fact(A, x-1);
    A[x-1][x-1] = result;
    int i;
    for(i = 0; i < x-1; i++){
        A[x-1][i] = A[x-2][i];
    }
    return result;

}
```



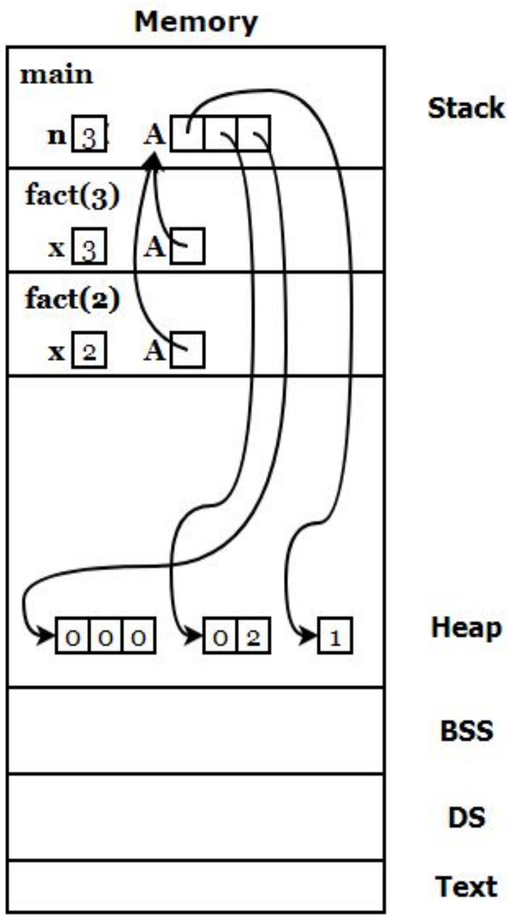
Example 2



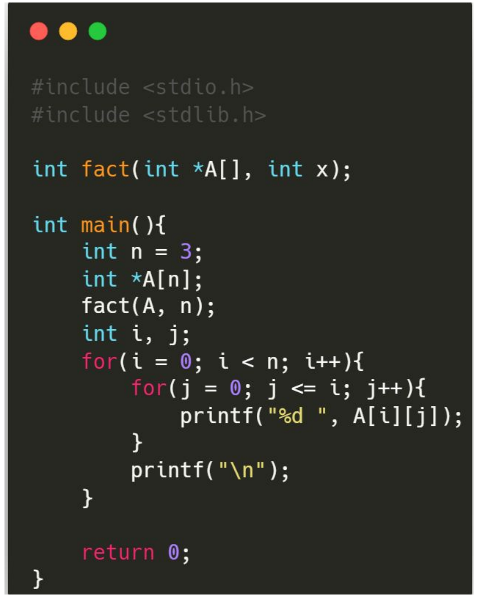
```
#include <stdio.h>
#include <stdlib.h>

int fact(int *A[], int x);

int main(){
    int n = 3;
    int *A[n];
    fact(A, n);
    int i, j;
    for(i = 0; i < n; i++){
        for(j = 0; j <= i; j++){
            printf("%d ", A[i][j]);
        }
        printf("\n");
    }

    return 0;
}
```



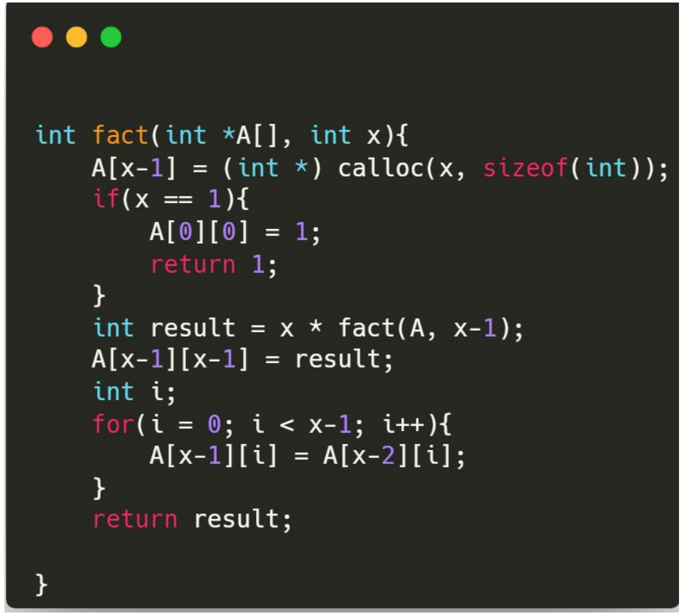
```
int fact(int *A[], int x){
    A[x-1] = (int *) calloc(x, sizeof(int));
    if(x == 1){
        A[0][0] = 1;
        return 1;
    }
    int result = x * fact(A, x-1);
    A[x-1][x-1] = result;
    int i;
    for(i = 0; i < x-1; i++){
        A[x-1][i] = A[x-2][i];
    }
    return result;

}
```



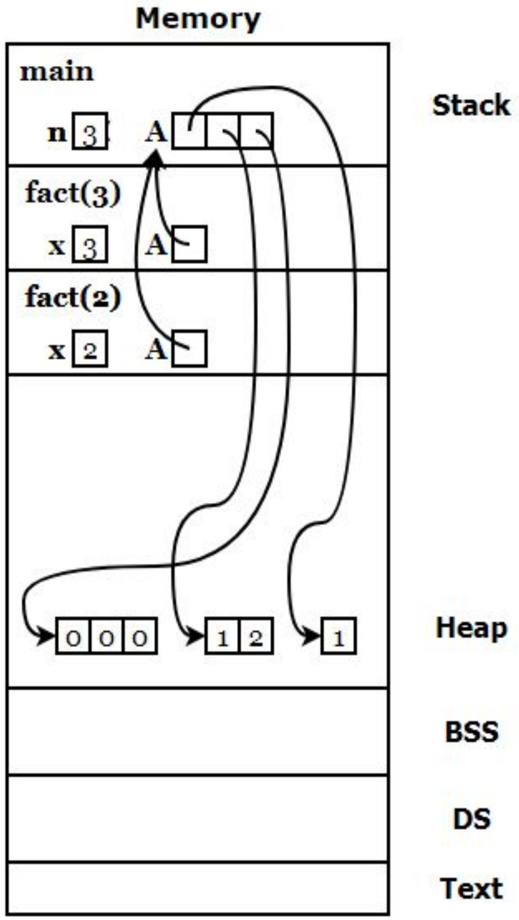
Example 2



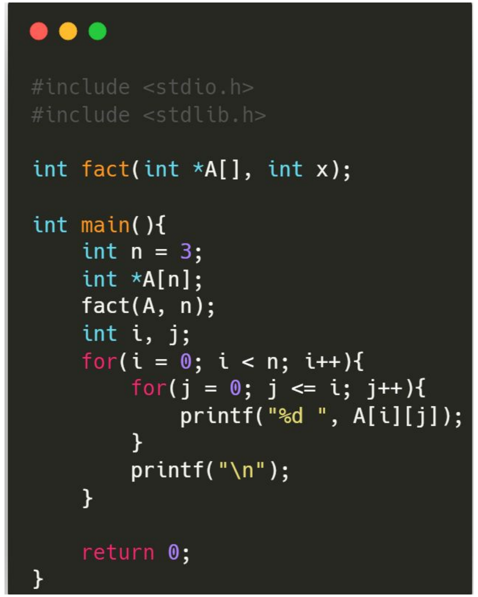
```c
#include <stdio.h>
#include <stdlib.h>

int fact(int *A[], int x);

int main(){
    int n = 3;
    int *A[n];
    fact(A, n);
    int i, j;
    for(i = 0; i < n; i++){
        for(j = 0; j <= i; j++){
            printf("%d ", A[i][j]);
        }
        printf("\n");
    }

    return 0;
}
```



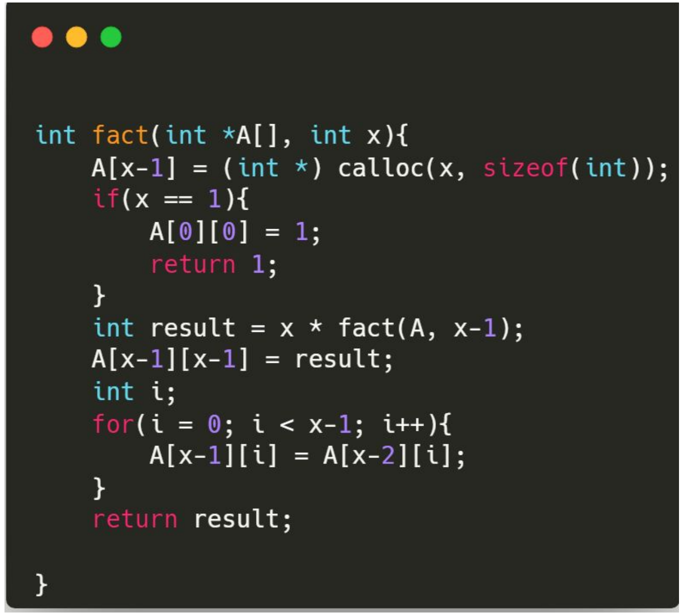
```c
int fact(int *A[], int x){
    A[x-1] = (int *) calloc(x, sizeof(int));
    if(x == 1){
        A[0][0] = 1;
        return 1;
    }
    int result = x * fact(A, x-1);
    A[x-1][x-1] = result;
    int i;
    for(i = 0; i < x-1; i++){
        A[x-1][i] = A[x-2][i];
    }
    return result;

}
```



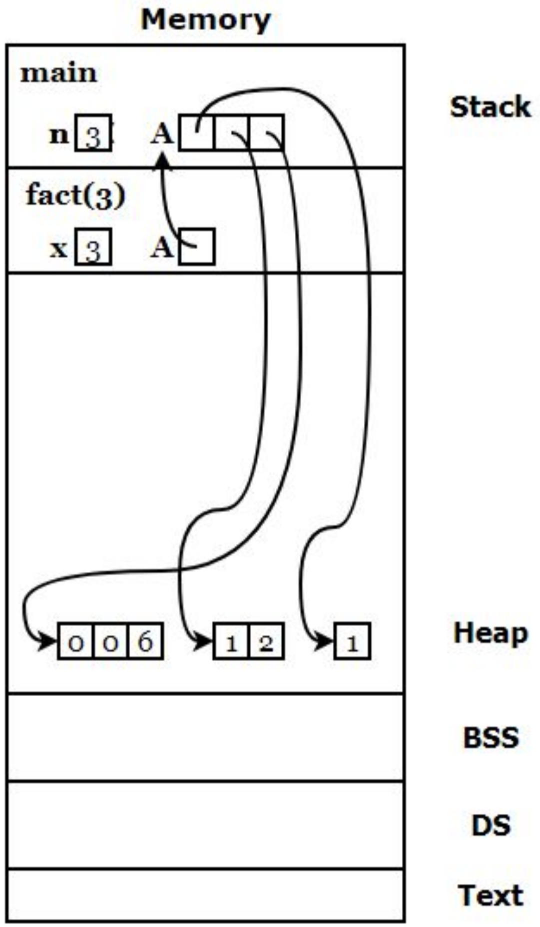
Example 2



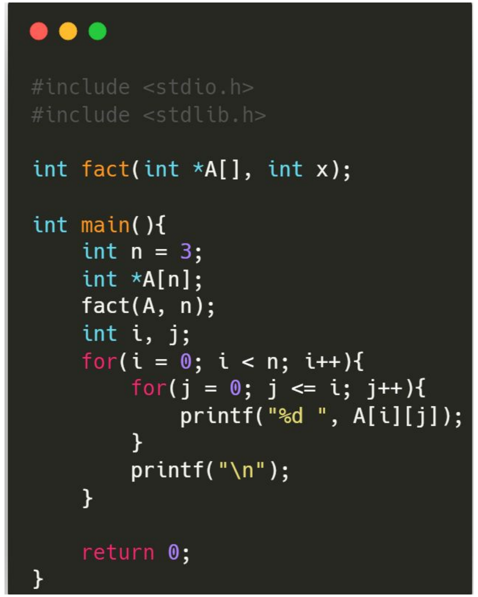
```
#include <stdio.h>
#include <stdlib.h>

int fact(int *A[], int x);

int main(){
    int n = 3;
    int *A[n];
    fact(A, n);
    int i, j;
    for(i = 0; i < n; i++){
        for(j = 0; j <= i; j++){
            printf("%d ", A[i][j]);
        }
        printf("\n");
    }

    return 0;
}
```



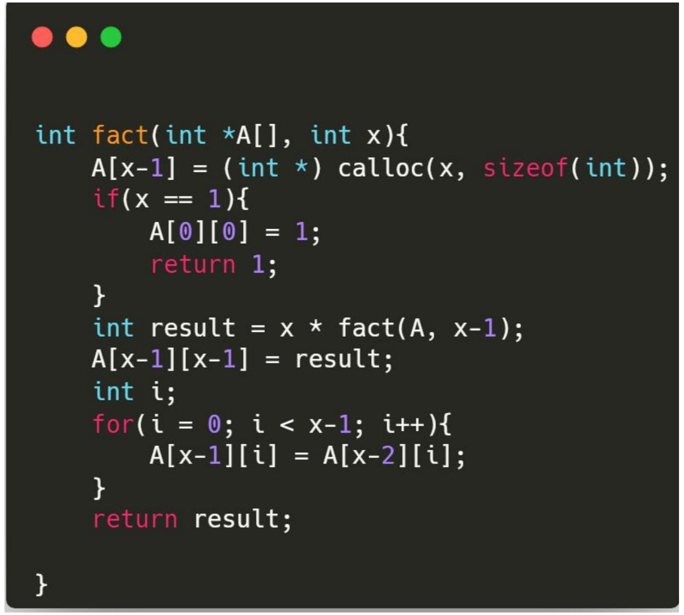
```
int fact(int *A[], int x){
    A[x-1] = (int *) calloc(x, sizeof(int));
    if(x == 1){
        A[0][0] = 1;
        return 1;
    }
    int result = x * fact(A, x-1);
    A[x-1][x-1] = result;
    int i;
    for(i = 0; i < x-1; i++){
        A[x-1][i] = A[x-2][i];
    }
    return result;

}
```



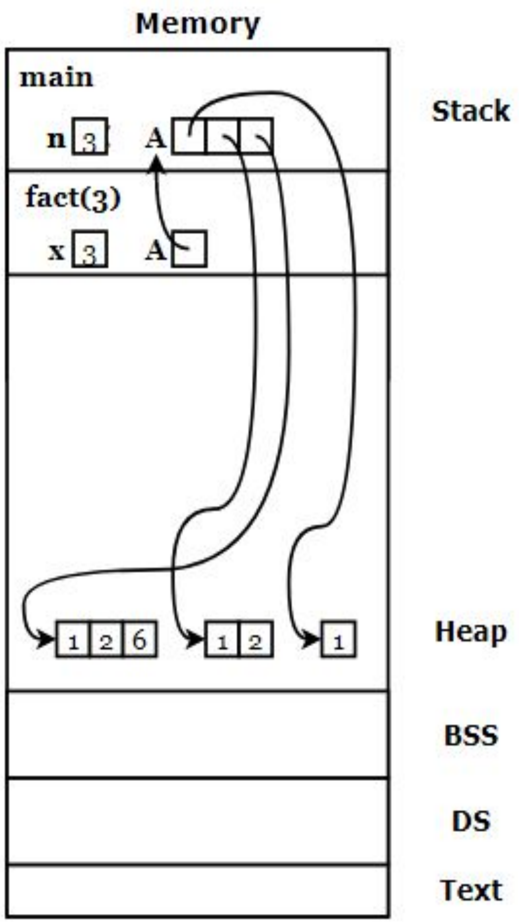
Example 2



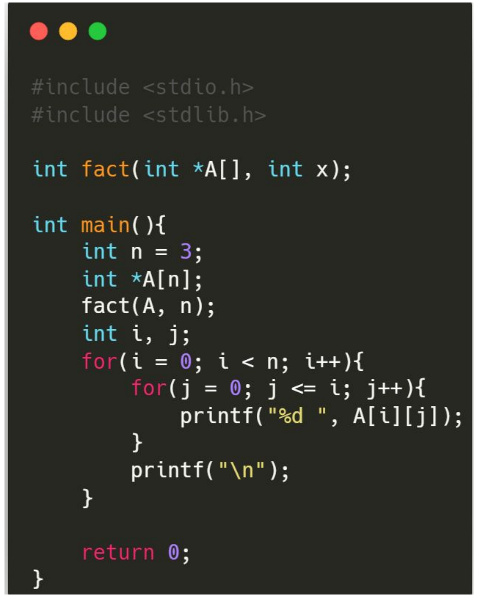
```c
#include <stdio.h>
#include <stdlib.h>

int fact(int *A[], int x);

int main(){
    int n = 3;
    int *A[n];
    fact(A, n);
    int i, j;
    for(i = 0; i < n; i++){
        for(j = 0; j <= i; j++){
            printf("%d ", A[i][j]);
        }
        printf("\n");
    }

    return 0;
}
```



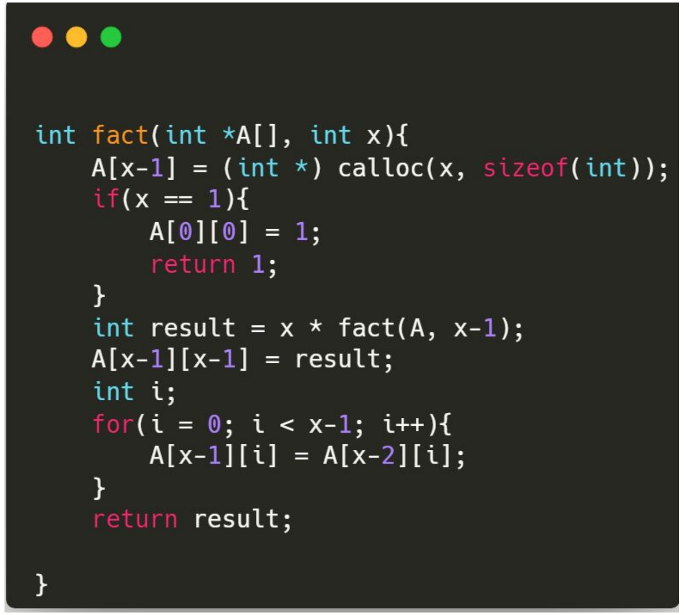
```c
int fact(int *A[], int x){
    A[x-1] = (int *) calloc(x, sizeof(int));
    if(x == 1){
        A[0][0] = 1;
        return 1;
    }
    int result = x * fact(A, x-1);
    A[x-1][x-1] = result;
    int i;
    for(i = 0; i < x-1; i++){
        A[x-1][i] = A[x-2][i];
    }
    return result;

}
```



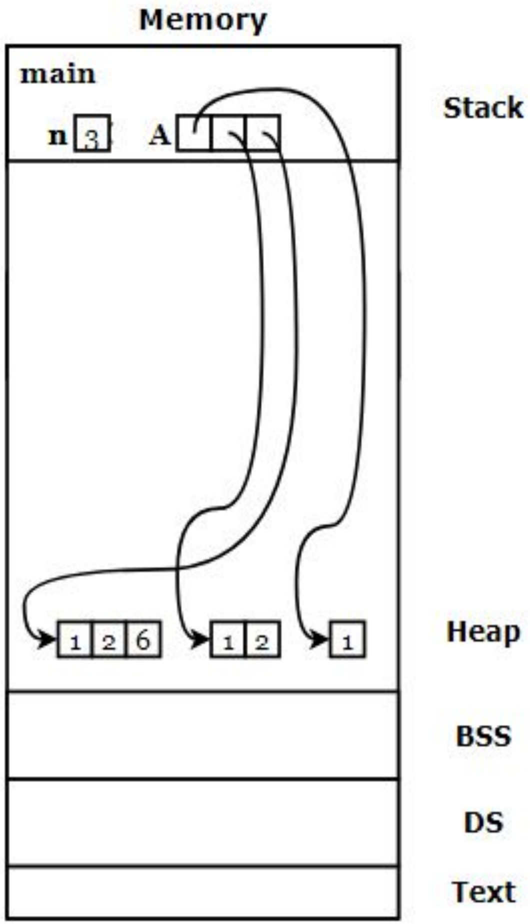
Example 2



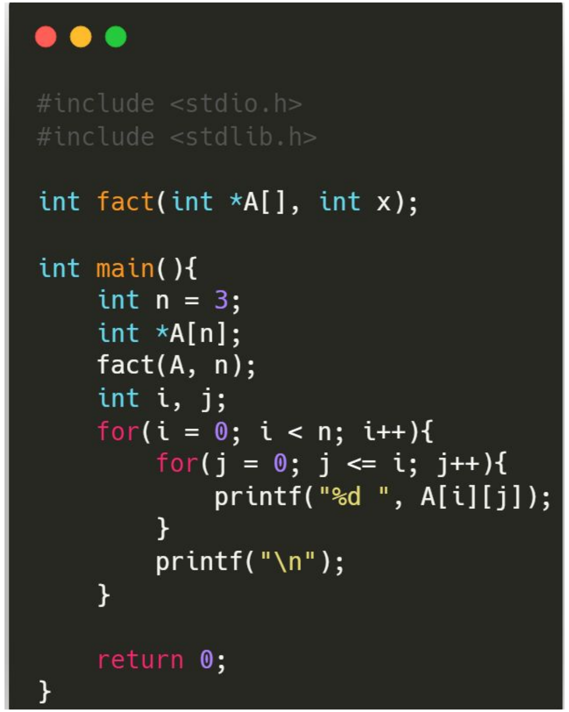
```
#include <stdio.h>
#include <stdlib.h>

int fact(int *A[], int x);

int main(){
    int n = 3;
    int *A[n];
    fact(A, n);
    int i, j;
    for(i = 0; i < n; i++){
        for(j = 0; j <= i; j++){
            printf("%d ", A[i][j]);
        }
        printf("\n");
    }

    return 0;
}
```



```
int fact(int *A[], int x){
    A[x-1] = (int *) calloc(x, sizeof(int));
    if(x == 1){
        A[0][0] = 1;
        return 1;
    }
    int result = x * fact(A, x-1);
    A[x-1][x-1] = result;
    int i;
    for(i = 0; i < x-1; i++){
        A[x-1][i] = A[x-2][i];
    }
    return result;

}
```

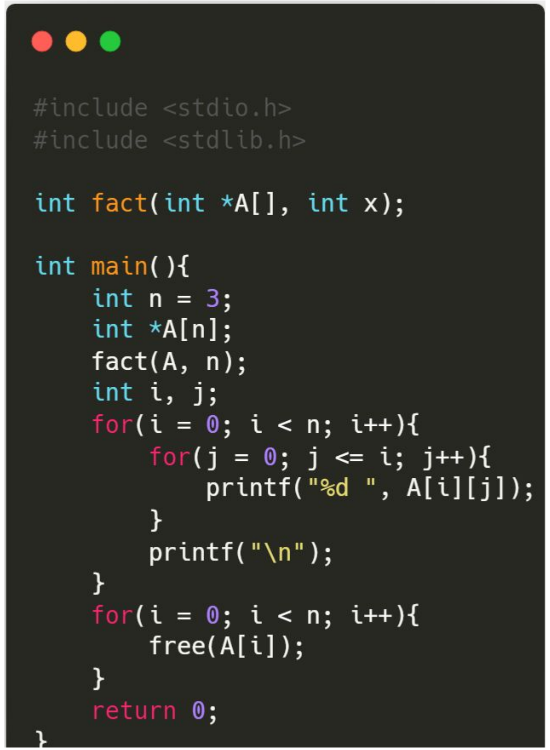


Example 2



```
#include <stdio.h>
#include <stdlib.h>

int fact(int *A[], int x);

int main(){
    int n = 3;
    int *A[n];
    fact(A, n);
    int i, j;
    for(i = 0; i < n; i++){
        for(j = 0; j <= i; j++){
            printf("%d ", A[i][j]);
        }
        printf("\n");
    }

    return 0;
}
```

Free memory no longer needed



```
#include <stdio.h>
#include <stdlib.h>

int fact(int *A[], int x);

int main(){
    int n = 3;
    int *A[n];
    fact(A, n);
    int i, j;
    for(i = 0; i < n; i++){
        for(j = 0; j <= i; j++){
            printf("%d ", A[i][j]);
        }
        printf("\n");
    }
    for(i = 0; i < n; i++){
        free(A[i]);
    }
    return 0;
}
```